LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-576314

Foundations of Feature Selection and Classification for Non-Gaussian Distributed Targets

G. A. Clark

August 23, 2012

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Foundations of Feature Selection and Classification for Non-Gaussian Distributed Targets

Grace A. Clark, Ph.D., IEEE Fellow
Visiting Research Professor
Department of Electrical and Computer Engineering
Naval Postgraduate School
Spanagel Hall, Room 464,
833 Dyer Road
Monterey, CA 93943-5121
831-656-3493 (Office)
gaclark@nps.edu

Lawrence Livermore National Laboratory
Contractor to the U.S. Department of Energy
7000 East Ave., L-130, Livermore, CA 94550
(925) 423-9759 (Office), (925) 422-2495 (FAX),
clark9@llnl.gov

August 24, 2012

LLNL-TR-576314

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Acknowledgments

The author gratefully acknowledges the technical discussions of this work with Ensign Matt Wilder, MSEE student at the Naval Postgraduate School (NPS), Monterey, CA, and Professor Monique Fargues of NPS. This work was supported by the Lawrence Livermore National Laboratory (LLNL) and the Naval Postgraduate School (NPS) during the author's Professional Research and Teaching Leave at NPS from March 29, 2010 through May 14, 2012.

Auspices:
This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Abstract

This report describes the author's research toward creating algorithms for selecting useful statistical features in pattern/target classification problems in which the features are non-Gaussian distributed. In engineering practice, it is common to either (1) Not perform any feature selection procedure, or (2) Use a feature selection algorithm that assumes the features are Gaussian distributed. This process can be far from optimal if the data are actually non-Gaussian, as they frequently are in practice. This research has the goal of mitigating that problem by creating a useful algorithm that can be used in practice.

The approach is to focus on the performance indices used in common feature selection algorithms, such as Branch and Bound algorithms and the Sequential Forward Selection Algorithm. Ordinarily, the performance index used to measure the class separation in feature space involves assuming the data are Gaussian and deriving tractable performance indices that use closed form equations that represent the probability density functions of the class data. The advantage of this approach is that it produces feature selection algorithms that have low computational complexity and do not require knowledge of the actual data densities. The disadvantage is that these algorithms may not perform reasonably when the data are non-Gaussian, as they commonly are in practice.

This research examines the use of information-theoretic class separability measures that can deal with the non-Gaussian case. In particular, this work shows that the Hellinger Distance (a divergence measure), has very desirable mathematical properties and can be useful for feature selection. The work in this paper employs a multivariate kernel density estimator along with the Hellinger Distance and an optimal subset selection algorithm to do feature selection.

This report (1) Defines the classification and feature selection problems, (2) Describes the continuous variable Hellinger distance in one dimension and discusses its mathematical properties in comparison with other distance measures, (3) Develops the discrete variable Hellinger distance expression for one and two dimensions, and shows that the discrete Hellinger distance for multiple dimensions is a straightforward extension, and (4) Presents experimental density estimation results using a new MATLAB implementation of a multivariate kernel density estimator.

The draft of this report served as a starting point for Ensign Matthew Wilder's MSEE thesis work at the Naval Postgraduate School (NPS) in 2010 and 2011. Ensign Wilder completed his thesis and graduated from NPS in June 2011 (M. J. Wilder, "*Automatic Target Recognition: Statistical Feature Selection of Non-Gaussian Distributed Target Classes*," MSEE Thesis, Naval Postgraduate School, Thesis Advisor: Grace A. Clark, Second Reader: Monique P. Fargues, June 2011). In the thesis, Ensign Wilder created implementations of the Branch and Bound and Sequential Feature Selection algorithms, extended the multivariate kernel density estimator code, and evaluated the performance of Hellinger Distance-based feature selection algorithms using both simulated data and data from a real-world benchmark data set. The new algorithm was shown to be very effective for both Gaussian and non-Gaussian data sets. The advantages gained for non-Gaussian data come at the cost of increased computational complexity. Using a laptop computer for practical problems, we generally use initial feature vectors containing no more than about ten or twelve features. Proposed future research lies in creating faster kernel density estimator algorithms that would allow for the use of higher dimensional data.

Contents

List of Figures

Chapter 1

Introduction

A block diagram for a generic automatic target recognition (ATR) system appears in Figure 1.1. This ATR system uses multiple sensors and sensor data fusion along with feature analysis and classification algorithms to discriminate (or classify) targets of interest that are embedded in the measurements. Feature extraction and selection are key to the success of the ATR system, because the classification results can be only as good as the input features.

This purpose of this report is to describe the author's research toward creating algorithms for selecting useful statistical features in pattern/target classification problems in which the features are non-Gaussian distributed. In engineering practice, it is common to either (1) Not perform any feature selection procedure, or (2) Use a feature selection algorithm that assumes the features are Gaussian distributed. This process can be far from optimal if the data are actually non-Gaussian. This research has the goal of mitigating that problem by creating a useful algorithm that can be used in practice.

In general, we have no prior knowledge about what features or how many features will be most effective in classifying a given data set. Therefore, we use our judgment to extract a relatively large number of features, fully expecting that some will be more useful than others. *Feature selection* is the process of choosing a subset of the large feature set for final use by the classifier. We do this by rank ordering the features according to their importance or value for classification, then choosing to use only the most important ones. We discuss quantitative measures of importance later in this section. Note that feature extraction and feature selection are usually not performed just one time each. They become part of an iterative process of feature analysis used to obtain a final feature set. However, once the feature set to be used is established, the features are inserted directly into the classifier without further feature selection. In some approaches, feature selection is performed in conjunction with the classifier during the training process.

Feature selection is important for several reasons. *First*, we wish to minimize the effects of the curse of dimensionality, in the sense that the classification computational complexity increases rapidly with the dimension of the feature vector. *Second*, we wish to use only features that add significant value to the quality of the classification results. Unimportant or redundant features add negative or zero value and should be removed. It is significant that human feature analysis experts generally produce classification results based upon a very small number of the most important attributes of a signal. In this section, we describe algorithms for evaluating the importance of features. *Third*, if too many features are used, the classifier performance can actually degrade. Statistical decision theory tells us that the probability of correct classification is an increasing function of the number of features provided, if the sample size is very large. For many applications, however, a relatively small number of training sets are available and estimation errors are no longer negligible. Empirical studies have shown that the probability of correct classification is not generally a monotonically increasing function of the number of features used. It generally increases up to a point at which it reaches a "knee"in the curve and begins decreasing, finally leveling off at a value less than the value at the knee [71, 48]. Clearly, our goal is to find the number of features corresponding to the knee in the curve. This behavior of the probability of correct classification (P_{CC}) motivates us to seek systematic

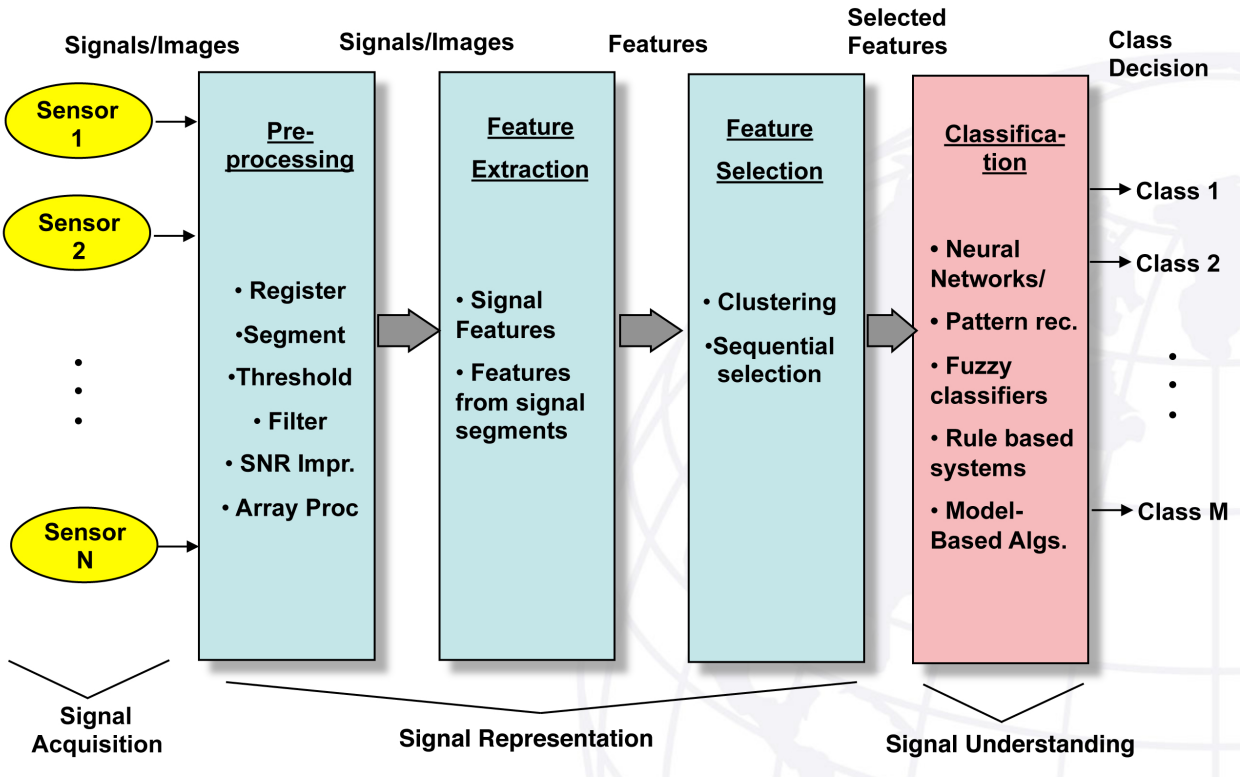


Figure 1.1: ATR Block Diagram: Measurements signals/images are processed to create a signal representation in terms of statistical features. These features are used by the classifier to make decisions in the form of target classifications.

feature selection algorithms such as the ones examined in this report. This is one reason why we are researching performance measures such as information theoretic divergences that obey the monotonicity property. This concept is examined in more detail in a later chapter. *Fourth*, an important by-product of feature selection can sometimes be increased knowledge of the physical processes that create the data. By understanding which features are statistically most important, we can often draw important conclusions about the physical reasons why they are important, and this can lead to productive insights that aid in the system design. *Fifth* , since we can use multiple sensors, we wish to use sensor feature fusion; so feature selection has the potential to determine which sensors are the most important.

Chapter 2

Bayesian Classification

In this chapter, we summarize the principles of Bayesian Classification theory for the two-class (or binar hypothesis) case [82, 84, 35, 85, 86]. Using general notation, we consider the problem in which the goal is to decide between two hypotheses, H_0 and H_1, as depicted in Figure (2.3). For example, if we are interested in detecting damage in electrical cables, we would let H_0 denote the null hypothesis, or the hypothesis in which the cable is not damaged. We let H_1 denote the hypothesis in which the cable is damaged. For other hypothesis testing problems, we use different definitions. For example for statistical whiteness testing, H_0 denotes the hypothesis in which the innovations are statistically white and H_1 denotes the hypothesis in which the innovations are not statistical white [3].

We measure classification performance using the Receiver Operating Characteristic (ROC) curve from communication theory [82, 84, 35, 85, 86]. The ROC curve is a plot of probability of detection vs. probability of false alarm. A statistical confidence interval should be calculated about the probability of correct classification for each point in the ROC curve, using the techniques described in a later section.

Building a ROC curve requires having an ensemble of statistical samples for hypothesis H_0 and a corresponding ensemble of statistical samples for hypothesis H_1. By creating a feature vector of length p, we create a p-dimensional feature space, in which the values of the features can be plotted or visualized. The classifier divides the feature space into classes (in this case, two classes), by creating a decision surface in feature space.

2.1 Hypothesis Testing

This section summarizes the concepts involved in constructing a detector/classifier for the binary hypothesis case [82, 84, 35, 85, 86]. The multiple hypothesis testing problem is a straightforward extension of the binary problem, and is described in [82, 84, 85].

Consider the decision problem in which a source of some kind generates an output that consists of one of two choices corresponding to two hypotheses H_0 and H_1. Each hypothesis maps to a point in observation space. The observation space corresponds to a set of N observations that can be denoted by the observation vector $\underline{X}$:

$$\underline{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad \text{(Observation Vector)} \tag{2.1}$$

The system generates observations according to two conditional probability densities $f(\underline{X}|H_0)$ and $f(\underline{X}|H_1)$. We know that either H_0 or H_1 is true and we are required to make a choice between them. Each time the experiment is

"Confusion Matrix" or Contingency Table for Binary Hypothesis Testing

Truth / Declaration	True Hypothesis H_0 (Null)	True Hypothesis H_1
Declared Hypothesis H_0(Null)	$P(H_0 \mid H_0) = P_{Spec} = Specificity$ $= \frac{\#\, H_0\ Samples\ \ Declared\ H_0}{\#\, H_0\ Samples}$	$P(H_0 \mid H_1) = P_{Miss} = P(Miss)$ $= \frac{\#\, H_1\ Samples\ \ Declared\ H_0}{\#\, H_1\ Samples}$
Declared Hypothesis H_1	$P(H_1 \mid H_0) = P_{FA} = P(False\ Alarm)$ $= \frac{\#\, H_0\ Samples\ \ Declared\ H_1}{\#\, H_0\ Samples}$	$P(H_1 \mid H_1) = P_D = P(Detection)$ $= \frac{\#\, H_1\ Samples\ \ Declared\ H_1}{\#\, H_1\ Samples}$
	$P(H_0 \mid H_0) + P(H_1 \mid H_0) = 1$	$P(H_0 \mid H_1) + P(H_1 \mid H_1) = 1$

$$P_{cc} = P(Correct\ Classification) = P(H_0 \mid H_0)P(H_0) + P(H_1 \mid H_1)P(H_1)$$
$$P_e = P(Error) = 1 - P_{cc} = P(H_0 \mid H_1)P(H_1) + P(H_1 \mid H_0)P(H_0)$$
$$\text{Assume: Correct classification is given zero cost} \Rightarrow C_{00} = C_{11} = 0$$
$$\text{Incorrect classification is given full cost} \Rightarrow C_{01} = C_{10} = 1$$

Figure 2.1: Confusion Matrix (Contingency Table). The two hypotheses are denoted H_0, the null hypothesis, and H_1. Note that for special case in which the prior probabilities are equal ($P(H_0) = P(H_1) = \frac{1}{2}$), the probability of correct classification becomes $P_{CC} = \frac{1}{2}[P_D + (1 - P_{FA})]$.

conducted, one of four events can occur: (1) H_0 is true and we declare H_0, (2) H_0 is true and we declare H_1, (3) H_1 is true and we declare H_1, (4) H_1 is true and we declare H_0. The first and third alternatives correspond to correct choices. The second and fourth alternatives correspond to errors. The purpose of a decision criterion is to assign relative importance to the four possibilities. The four possibilities and their associated probabilities are depicted in the Contingency Table, or Confusion Matrix shown in Figure (2.1). Note that the four cases correspond to the Probability of Detection $P(H_1|H_1)$, Probability of a Miss $P(H_0|H_1)$, Specificity $P(H_0|H_0)$, and the Probability of False Alarm $P(H_1|H_0)$. The way to read these probabilities is demonstrated for the example of Probability of False Alarm $P(H_1|H_0)$. This should be read "the probability that the classifier declares hypothesis H_1 to be true, given that hypothesis H_0 is actually true." The figure shows how these probabilities are related to each other. It also defines the Probability of Error and the Probability of Correct Classification, which are discussed later.

2.2 The Bayes Decision Rule

The Bayes test assumes that there exist prior probabilities (priors) for the hypotheses and costs associated with the four courses of action. The priors $P(H_0)$ and $P(H_1)$ represent information available about the source prior to conducting the experiments. The costs for the four possible courses of action are given by C_{00}, C_{10}, C_{11} and C_{01},

where C_{ij} is the cost of deciding H_i given that H_j is true. Once the costs have been assigned, the decision rule is based on minimizing the expected cost, which is known as the Bayes risk $\mathfrak{R}$:

$$\mathfrak{R} = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij} P(H_i|H_j)P(H_j) \qquad \text{(Bayes Risk)} \tag{2.2}$$

We assume throughout this discussion that the cost of an incorrect decision is higher than the cost of a correct decision. In other words, $C_{10} > C_{00}$ and $C_{01} > C_{11}$. Under this assumption, the detector that minimizes the Bayes risk is given by the following:

$$\frac{f(\underline{X}|H_1)}{f(\underline{X}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P(H_0)(C_{10}-C_{00})}{P(H_1)(C_{01}-C_{11})} \qquad \text{(Bayes Decision Criterion)} \tag{2.3}$$

The ratio of the conditional densities is called the likelihood ratio and is denoted by $\Lambda(\underline{X})$:

$$\Lambda(\underline{X}) = \frac{f(\underline{X}|H_1)}{f(\underline{X}|H_0)} \qquad \text{(Likelihood Ratio)} \tag{2.4}$$

Because this is a ratio of two functions of a random variable, the likelihood ratio is a random variable. A very important result is that regardless of the dimensionality of the observations $\underline{X}$, the likelihood ratio $\Lambda(\underline{X})$ is a one-dimensional variable. This idea is of fundamental importance in hypothesis testing. Regardless of the dimension of the observation space, the decision space is one-dimensional. The quantity on the right hand side of the relation 2.3 is the threshold of the test and is denoted by η:

$$\eta \triangleq \frac{P(H_0)(C_{10}-C_{00})}{P(H_1)(C_{01}-C_{11})} \qquad \text{(Threshold)} \tag{2.5}$$

Thus, the Bayes criterion leads to a likelihood ratio test (LRT):

$$\Lambda(\underline{X}) \underset{H_0}{\overset{H_1}{\gtrless}} \eta \qquad \text{(Likelihood Ratio Test)} \tag{2.6}$$

We see that the test threshold allows for weighting according to the priors and the costs. This allows the user flexibility in choosing a threshold that is best for the problem at hand.

Key Property of the Likelihood Ratio Test: Regardless of the observation space (the size of the observation vector $\underline{X}$, the Bayes decision test consists of comparing a scalar variable $\Lambda(\underline{X})$ with a scalar threshold η. This is a very powerful result, as it allows us to collapse a problem in a large vector space to a simpler problem in scalar space.

Another fundamental concept is that of a sufficient statistic $l(\underline{X})$. This is a function of the observed data which has the property that $\Lambda(\underline{X})$ can be written as a function of $l(\underline{X})$. This means that when making a decision, knowing the value of the sufficient statistic is just as good as knowing $\underline{X}$. When we choose a sufficient statistic, we are simply describing each point in the coordinate space of the observations that is more useful for the decision problem. This concept allows us to use a coordinate system that suits our problem best in a practical sense [82].

Using the Bayes decision rule, we can now generate a receiver operating characteristic (ROC) curve that depicts the tradeoff between probability of detection and probability of false alarm (see Figure (2.3). Details of ROC curve development are given in the next section.

2.2.1 The Receiver Operating Characteristic (ROC) Curve

The Receiver Operating Characteristic (ROC) curve was developed in communications theory as a tool for evaluating the performance of a classifier [82, 83]. The ROC curve is extremely powerful and widely used in many

technical areas of study. The ROC curve is defined by plotting Probability of False Alarm on the abscissa and Probability of Detection on the ordinate. Note that because particular pairs of probabilities in Figure (2.1) sum to one, it can be shown that an equivalent way of defining a ROC curve would be to plot Probability of a Miss vs. Specificity.

There are two general ways in which a ROC Curve can be constructed, as depicted in Figure (2.2). If one is given closed-form expressions for the conditional probability density functions, then one can integrate them as in Figure (2.3) and (2.4). If one is given ensembles of finite numbers of observation data corresponding to the hypotheses, then a contingency table can be constructed as in Figure (2.1). The probability of detection and probability of false alarm are taken directly from the contingency table. Or, if estimates of the pdf's are available, they can be integrated to obtain the probabilities, as in Figure (2.3) and (2.4).

Example of the Construction of a ROC curve: Using general notation, we consider the two-class hypothesis testing problem in which the goal is to decide between two hypotheses, H_0 and H_1, as depicted in Figure (2.3). The problem depicted is that of detecting a signal in stochastic noise. The algorithm chosen determines the detection statistic r on which we place a variable threshold r_0. When the threshold is varied across the values of r, a curve is traced out with a shape as shown in Figure (2.3). At each value of the threshold, a point on the ROC curve is determined.

Minimax Operating Point on the ROC Curve: Referring to Figure (2.3), the goal of our work is to find the threshold γ at which the classifier achieves the minimax operating point. This is the point, often called the "knee " of the curve, at which P_{FA} is minimum and P_D is maximum. Van Trees [82] derives the theoretical minimax operating point. In practice, the author computes the probability of correct classification P_{CC} for all of the range of values of the threshold. The desired minimax operating point occurs where the threshold maximizes P_{CC}.

Families of ROC Curves: Families of ROC curves can be formed by varying the signal-to-noise ratio (SNR) or other appropriate quantities of interest, as depicted in Fig (2.3). Intuitively, we see that if the SNR is high (the two pdf's are easily separable), then the ROC curve moves toward the axes and the $(0, 1)$ point. If the SNR low, then the ROC curve moves toward the "45 degree "line that connects the point $(0, 0)$ with $(1, 1)$. Theoretically, the ROC curve should never fall below this line.

This concept leads to commonly-used curves that are built from combinations of the three basic quantities P_D, P_{FA} and SNR. For example, people sometimes plot P_D vs. SNR and let P_{FA} vary. A very nice practical exposition of ROC curves is given in [83].

2.2.2 Bayes Detection Rule Using Conditional Posterior Probabilities

Another equivalent way to view the Bayes likelihood ratio test is to rewrite it in terms of conditional posterior probabilities. This formulation can be useful in some applications [12]. For this derivation, we assume that the costs of classification errors are the same for both hypotheses. Under this assumption, the conditional posterior probability $P(H_1|\underline{X})$ for the binary hypothesis case can be written as follows [82].

$$P(H_1|\underline{X}) = \frac{f(\underline{X}|H_1)P(H_1)}{f(\underline{X}|H_0)P(H_0) + f(\underline{X}|H_1)P(H_1)} \qquad \text{(Posterior Probability)} \qquad (2.7)$$

Similarly, we can write the conditional posterior $P(H_0|\underline{X})$ for the binary hypthesis case can be written as follows [82]:

$$P(H_0|\underline{X}) = \frac{f(\underline{X}|H_0)P(H_0)}{f(\underline{X}|H_0)P(H_0) + f(\underline{X}|H_1)P(H_1)} \qquad \text{(Posterior Probability)} \qquad (2.8)$$

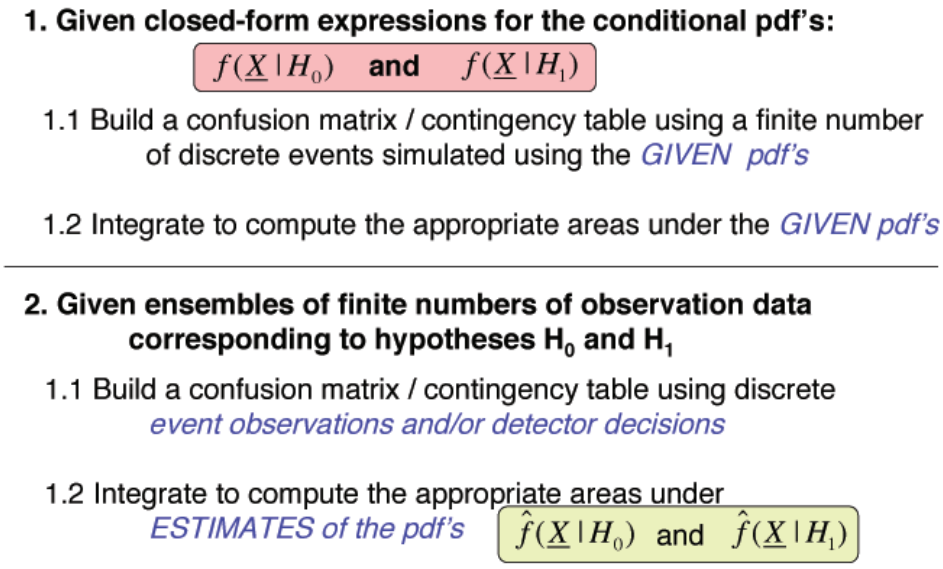


Figure 2.2: There are two general ways in which a ROC Curve can be constructed. If one is given closed-form expressions of the conditional probability density functions, then one can integrate them as in Figure (2.3) and (2.4). If one is given ensembles of finite numbers of observation data corresponding to the hypotheses, then a contingency table can be constructed as in Figure (2.1). The probability of detection and probability of false alarm are taken directly from the contingency table. Or, if estimates of the pdf's are available, they can be integrated to obtain the probabilities, as in Figure (2.3) and (2.4).

One can show that the Bayes decision rule can take the following equivalent form:

$$P(H_1|\underline{X}) \underset{H_0}{\overset{H_1}{\gtrless}} .5 \qquad \text{(Posterior Probability Test} \tag{2.9}$$

Using this equivalent Bayes decision rule, we can now generate a receiver operating characteristic (ROC) curve that depicts the tradeoff between probability of detection and probability of false alarm (see Figure (2.3) and (2.4). Details of ROC curve development are given in the next section. Let us now consider some important special cases of the Bayes decision rule.

2.2.3 Bayes Risk Special Case When $C_{00} = C_{11} = 0$ and $C_{01} = C_{10} = 1$

An important special case of the Bayes criterion is that in which a correct classification is assigned zero cost and an incorrect classification is assigned full cost. In this case, we assign $C_{00} = C_{11} = 0$ and $C_{01} = C_{10} = 1$. Inserting these values in the Bayes Risk of Equation (2.2), we obtain:

$$\begin{aligned} \Re = P(\text{Error}) &= P(H_0, H_1) + P(H_1, H_0) && (2.10)\\ &= P(H_0|H_1)P(H_1) + P(H_1|H_0)P(H_0) && (2.11) \end{aligned}$$

This version of the Bayes risk is called probability of error, and is a very effective criterion for evaluating detector/classifier performance [82, 84, 35, 85, 86]. Alternatively, we often use the fact that $P(\text{Error})+P(\text{Correct Classification}) = 1$ and define the probability of correct classification:

$$\begin{aligned} P(\text{Correct Classification}) = P(CC) &= P(H_1, H_1) + P(H_0, H_0) && (2.12)\\ &= P(H_1|H_1)P(H_1) + P(H_0|H_0)P(H_0) && (2.13) \end{aligned}$$

In this work, we use probability of correct classification as a very useful performance measure.

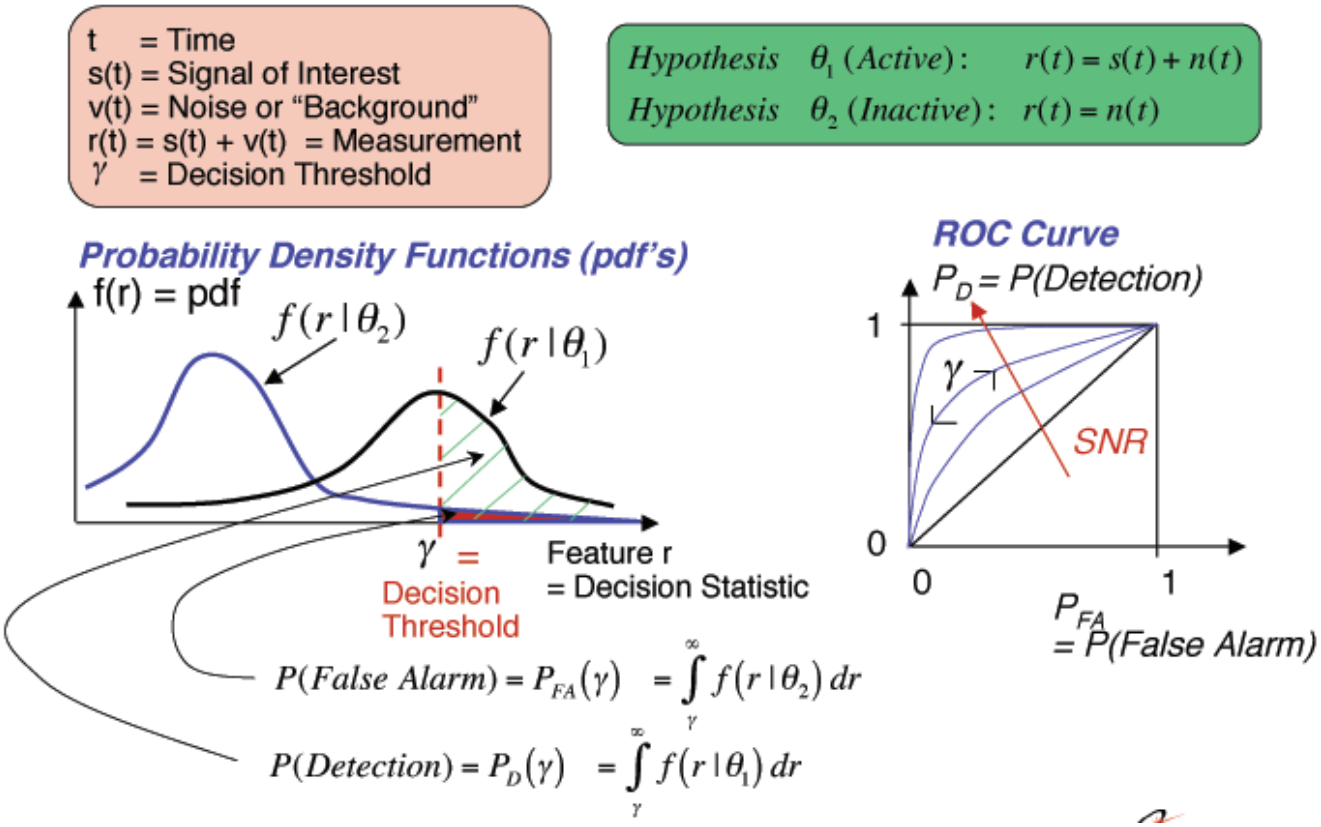
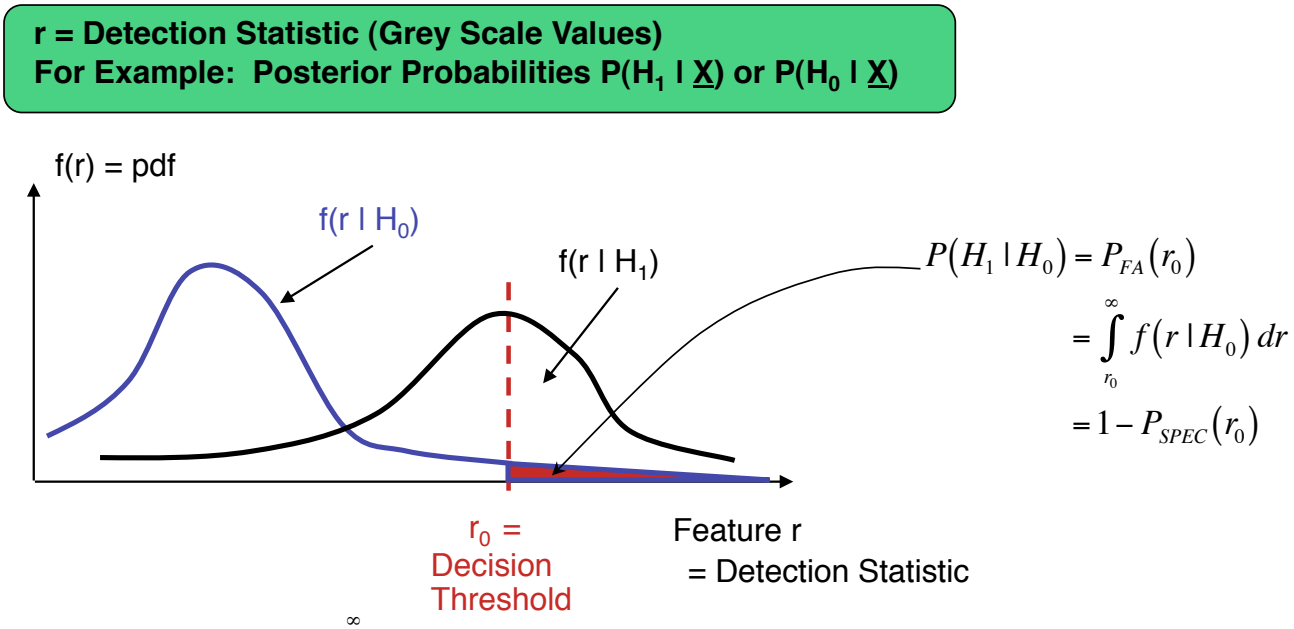


Figure 2.3: The ROC curve can be generated by integrating under the conditional probability density functions to compute P_D and P_{FA}. As the decision threshold is varied, a curve is mapped out in the ROC curve. Families of curves can be developed by letting the signal-to-noise (SNR) ratio vary.



$$P(H_1 \mid H_1) = P_D(r_0) = \int_{r_0}^{\infty} f(r \mid H_1)\, dr = 1 - P(H_0 \mid H_1) = 1 - P_{MISS}(r_0)$$

$$P(H_0 \mid H_1) = P_{MISS}(r_0) = \int_{-\infty}^{r_0} f(r \mid H_1)\, dr = 1 - P(H_1 \mid H_1) = 1 - P_D(r_0)$$

$$P(H_0 \mid H_0) = P_{SPEC}(r_0) = \int_{-\infty}^{r_0} f(r \mid H_0)\, dr$$

Figure 2.4: The details of the integrations under the ROC curve are depicted in this figure. Let r denote the test statistic of interest. Then the various conditional probabilities in the contingency table are computed by the integrations shown.

2.2.4 Probability of Correct Classification for the Special Case When Priors are Equal

Often in practice there exists insufficient information about an experiment to allow assignment of prior probabilities P_0 and P_1. In this case, it is common to assume that the priors are equal, so $P_0 = P_1 = 1/2$. Under this condition, the probability of correct classification becomes

$$P(CC) = \frac{1}{2}[P(H_1|H_1) + P(H_0|H_0)] \tag{2.14}$$

This can now be written in terms of the probability of detection and probability of false alarm as follows:

$$P(CC) = \frac{1}{2}[P_D + (1 - P_{FA})] \tag{2.15}$$

2.2.5 The Bayes Minimax Test

A Bayes test designed to minimize the maximum possible risk is called a minimax test. For any choice of decision regions the risk expression can be written in terms of the quantities in Figures (2.3) and (2.4) as follows:

$$\Re = P(H_0)C_{10} + P(H_1)C_{11} + P(H_1)(C_{01} - C_{11})P_M - P(H_0)(C_{10} - C_{00})(1 - P_{FA}) \tag{2.16}$$

where $P_M = 1 - P_D$ is the probability of a miss. To minimize the maximum risk, we use a Bayes test designed assuming $P(H_1) = 1$. This implies that the coefficient of $P(H_1)$ must be zero, and the resulting minimax equation is:

$$(C_{11} - C_{00}) + (C_{01} - C_{11})P_M - (C_{10} - C_{00})P_{FA} = 0 \tag{2.17}$$

Assuming that $C_{00} = C_{11} = 0$ and substituting $P_M = 1 - P_D$, the risk becomes

$$\Re = P(H_0)C_{10}P_{FA} + P(H_1)C_{01}(1 - P_D) \tag{2.18}$$

and the minimax equation is

$$C_{01}(1 - P_D) = C_{10}P_{FA} \tag{2.19}$$

2.3 Statistical Confidence Interval on the Probability of Correct Classification

We are very interested in knowing the confidence with which we can specify the performance of classifiers. In this section, we present algorithms for computing confidence criteria.

In the process of evaluating classification/detector performance, we estimate conditional probabilities based upon experiments with real world data and a finite number of statistical samples. We can specify the performance in terms of sensitivity and specificity. In order to specify the performance fully, however, it is desired to specify the confidence we have in the estimates of the conditional probabilities. We can do this by calculating a statistical confidence interval about the probability of correct classification.

The classifier/detector performs a random experiment, the outcome of which can be classified in one of two mutually exclusive and exhaustive ways: success or failure. Success means that the classification is correct. Failure means that the classification is incorrect. Let N equal the number of independent trials. Let $p = P(CC) =$ the probability of success. Assume that p is the same on each repetition. Let $q = 1 - p =$ probability of failure. Now, let X_i be a random variable with $i = 1, 2, \ldots, N$ and

$$X_i = \begin{cases} 0, & \text{if the outcome of the ith trial is failure} \\ 1, & \text{if the outcome of the ith trial is success} \end{cases}$$

So, we can write $P\{X_i = 1\} = p$ and $P\{X_i = 0\} = 1 - p = q$.

Now, let Y be the sum of successes throughout N repetitions of the experiment.

$$Y = \sum_{i=1}^{N} X_i \tag{2.20}$$

Let $y \in \{y : y = 1, 2, \ldots, N\}$. Then, $Y = y$ iff exactly y of the variables $\{X_1, X_2, \ldots, X_N\}$ have value 1, and the remaining $N - y$ variables equal zero. There are $\binom{N}{y}$ ways in which exactly y ones can be assigned to y of the variables $X_1, X_2, \ldots, X_N$. Since $X_1, X_2, \ldots, X_N$ are mutually stochastically independent, the probability of each of these ways is $p^y(1-p)^{N-y}$. Now, $P(Y = y)$ is the sum of the probabilities of these $\binom{N}{y}$ mutually exclusive events, so

$$P(Y = y) = \begin{cases} \binom{N}{y} p^y (1-p)^{N-y} & , y = 0, 1, 2, \ldots, N \\ 0 & , \text{Elsewhere} \end{cases}$$

This is the probability density function (pdf) of a binomial distributed random variable, so Y is binomial distributed $b(N, p)$. Recall that $\binom{N}{y} = \frac{N!}{y!(n-y)!} = 0$ if $y > N$ or if $y < 0$. Also, note that for a binomial distribution, we have the mean $E\{y\} = Np$, and the variance $\sigma_y^2 = Npq$ [80, 89].

Now for the classification problem in which we conduct an experiment, we can calculate the quantities in the confusion matrix (or contingency table). The maximum likelihood estimate of p is given by:

$$\widehat{p} = \frac{\text{Number of Correct Classifications}}{\text{Number of Test Cases}}$$

We can also write this as follows:

$$\widehat{p} = \frac{y}{N} \tag{2.21}$$

$$\widehat{q} = 1 - \widehat{p} \tag{2.22}$$

We can write the confidence interval about the true value of p as follows, where α is the significance of the test.

$$P\{L < p < U\} = 1 - \alpha \tag{2.23}$$

where L and U are the lower and upper bounds, respectively, of the confidence interval. There are various ways to interpret the meaning of a confidence interval.

(1) This most common interpretation is to read the confidence interval relation above as follows: "With confidence $1 - \alpha$, the true p lies between L and U." However, this interpretation is not generally supported by statistical rigor. The preferred interpretations are given next.

(2) The classical interpretation is to read the confidence interval relation above as follows: "Prior to the repeated independent performances of the random experiment, the probability is $1 - \alpha$ that the random interval (L, U) includes the unknown fixed point (parameter) p [89]."

(3) The frequentist interpretation is to read the confidence interval relation as follows: "The confidence interval is a random interval that covers the true probability with frequency $1 - \alpha$. " This does not mean that a particular interval contains the true value of p with probability $1 - \alpha$ percent. The reason for this somewhat convoluted interpretation is that p is an unknown constant and not a random variable, and we cannot make probability statements about constants within the frequentist framework of statistics.

For the large sample size case ($N \geq 30$), the Gaussian approximation to the distribution of p can be used, and

the 95% confidence interval ($\alpha = .05$) is given by [89]:

$$L = \widehat{p} - 1.96\sqrt{\frac{\widehat{p}\,(1-\widehat{p})}{N}} \qquad \text{and} \tag{2.24}$$

$$U = \widehat{p} + 1.96\sqrt{\frac{\widehat{p}\,(1-\widehat{q})}{N}} \tag{2.25}$$

We can evaluate L and U, and plot them versus p and $\widehat{p}$, for various values of N, as in Figure (2.5). The plot is very instructive in showing how the confidence interval tightens as the sample size increases.

In many applications however, we are limited to small sample sizes, so we are forced to use more accurate estimates of L and U which are valid for small sample sizes. These estimates are given as follows [89]:

$$L = N\widehat{p} + 2 - 2\sqrt{\frac{N\widehat{p}\,(1-\widehat{p})+1}{N+4}} \qquad \text{and} \tag{2.26}$$

$$U = N\widehat{p} + 2 + 2\sqrt{\frac{N\widehat{p}\,(1-\widehat{p})+1}{N+4}} \tag{2.27}$$

For the small sample size case, we can evaluate L and U, and plot them versus p and $\widehat{p}$, for various values of N, as in Figure (2.6). Again, the plot is very instructive in showing how the confidence interval tightens as the sample size increases.

2.4 Confidence Intervals Based on the Bootstrap

There exist some relatively new algorithms in the literature for coping with small sample size statistics in signal processing applications. These involve exploiting the concept of the "bootstrap" algorithm to compute the confidence interval on the sample mean when the sample size is small [73, 74, 75, 76]. The author implemented an algorithm to compute the bootstrap mean and confidence interval on the mean. It was applied to a real-world problem in [3].

The bootstrap was introduced by Efron [75] as an approach to estimate confidence intervals for statistical parameters in circumstances in which standard methods cannot be applied. An example is when the statistical sample size is small because only a small number of data samples is available. The bootsrtrap does what the experimenter would do if it were possible - repeat the experiment many times to create a large sample size. The bootstrap approximates a large sample size by reassigning the observations randomly and recomputing key quantities many times (often thousands of times). These re-computations are then treated as repeated experiments.

The key to the bootstrap is very simple - resampling with replacement. Let $\mathcal{U}$ represent a finite population of individual units $U_1, U_2, \ldots, U_N$, any one of which is equally likely (with probability $1/N$) to be selected in a single random draw. A *random sample* of size n is defined to be a collection of n units $u_1, u_2, \ldots, u_n$ selected at random from $\mathcal{U}$.

In principle, the random sampling process uses a pseudo-random number generator to select independently a set of integers $j_1, j_2, \ldots, j_n$, each of which equals any value between 1 and N with probability $1/N$. These integers serve as indices that determine which members of $\mathcal{U}$ are selected in the random sample, $u_1 = U_{j1}, u_2 = U_{j2}, \ldots, u_n = U_{jn}$.

If *sampling with replacement* is used, then every sample is returned to the to the data set after sampling. This means that a single unit U_i is allowed to appear more than once, and some units may not appear at all. If *sampling without replacement* were used, then we would insist that the integers $j_1, j_2, \ldots, j_n$ be distinct. For the bootstrap, we use *sampling with replacement*.

For computing a bootstrap confidence interval on the mean of a random variable of interest, we let $\mathcal{X}$ be the random sample vector of samples in an ensemble used to compute the ensemble average. We then use the following procedure to compute the bootstrap confidence interval on the ensemble mean signal:

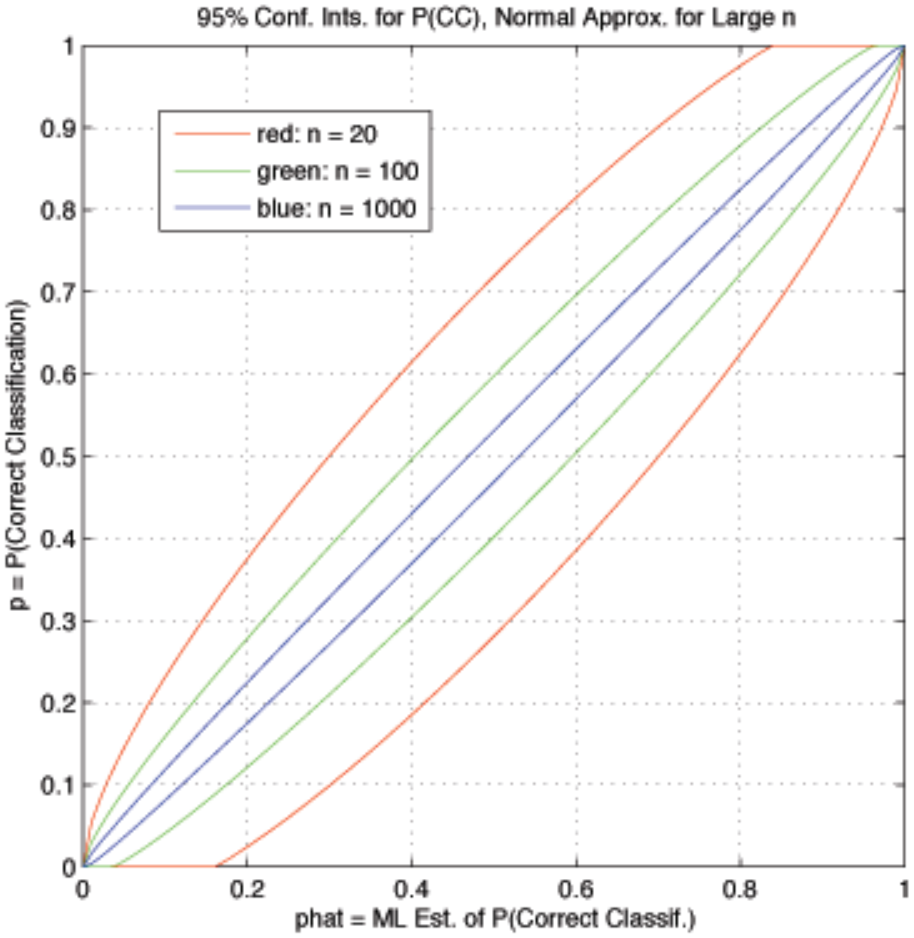


Figure 2.5: Large Sample Size Case: The 95 Percent confidence interval bounds (L, U) for the probability of correct classification are plotted, given the sample size n. Here, we use the Normal approximation ordinarily used for the the case in which n is large. Note how the confidence interval tightens as the sample size increases.

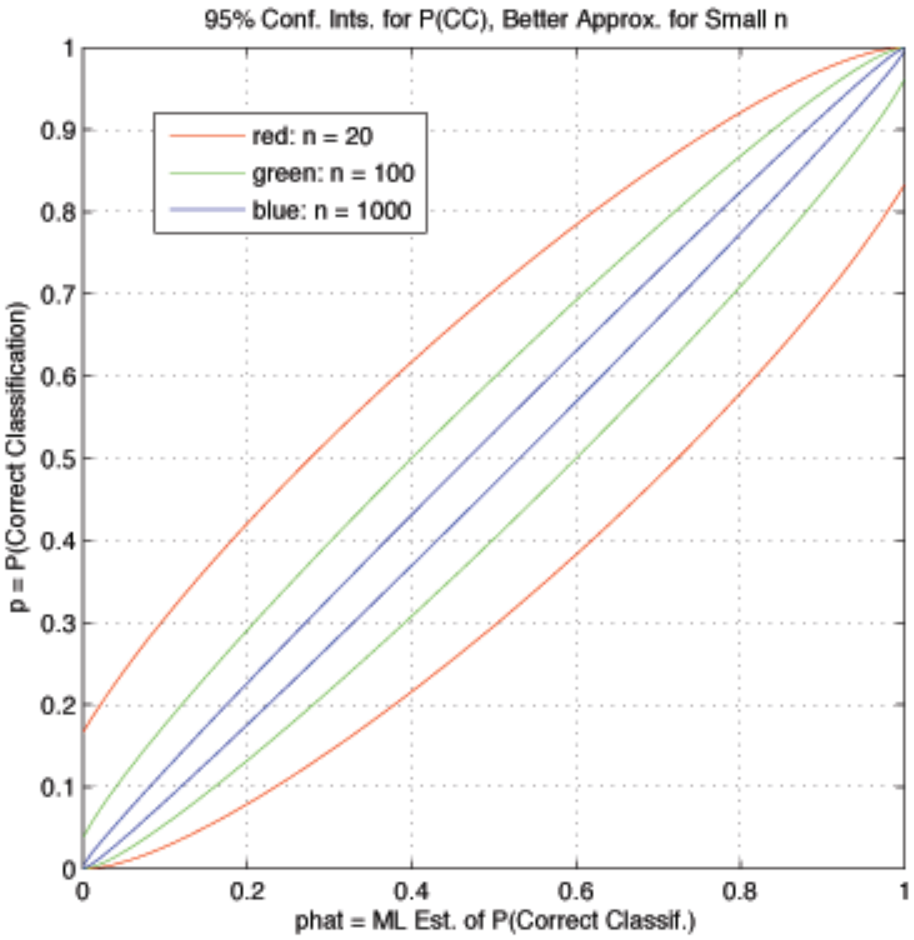


Figure 2.6: Small Sample Size Case: The 95 Percent confidence interval bounds (L, U) for the probability of correct classification are plotted, given the sample size n. Here, we use the better approximation for the the case in which n is small. Note how the confidence interval tightens as the sample size increases.

Algorithm for Estimating the Bootstrap Confidence Interval About the Mean for a Random Sample Vector:

Step 0, Conduct the Experiment:
Let the $N \times 1$ random sample vector be $\mathcal{X}$ as defined above.

Step 1, Resampling
Using a pseudo-random number generator, draw a random sample of N values with replacement from $\mathcal{X}$. We call this our bootstrap resample $\mathcal{X}^*$. Keep in mind that some of the original samples may appear more than once in $\mathcal{X}$ and others may appear not at all.

Step 2, Calculation of the Bootstrap Estimate of the Mean
Calculate a desired function of the of the values of $\mathcal{X}^*$. In our case, calculate the sample mean of the values in $\mathcal{X}^*$. Call this the bootstrap sample mean $\widehat{\mu}_1^*$.

Step 3, Repetition
Repeat Steps 1 and 2 a large number of times (for example, let $n = 1000$) to obtain a total of n bootstrap esimtates of the the mean: $\widehat{\mu}_1^*, \widehat{\mu}_2^*, \ldots, \widehat{\mu}_n^*$.

Step 4, Approximation of the Distribution of $\widehat{\mu}$.
Sort the bootstrap estimates of the means into increasing order to obtain the vector $\widehat{\mu}_{(1)}^*, \widehat{\mu}_{(2)}^*, \ldots, \widehat{\mu}_{(n)}^*$, where $\widehat{\mu}_{(k)}^*$ is the kth smallest of $\widehat{\mu}_1^*, \widehat{\mu}_2^*, \ldots, \widehat{\mu}_n^*$. If desired, one can compute a probability density function (pdf) estimate for $\widehat{\mu}$ and plot it for visual inspection. This step is not necessary, but it is quite instructive. For example, one could calculate a histogram or a kernel estimate of the pdf.

Step 5, Compute the Confidence Interval
The desired $(1-\alpha)100\%$ bootstrap confidence interval is given by the set of lower and upper bounds $[L, U] = \left[\widehat{\mu}_{(q_1)}^*, \widehat{\mu}_{(q_2)}^*\right]$, where α is the significance of the test, the index on the lower bound is $q_1 = \lfloor N\alpha/2 \rfloor$, which is the integer part of $N\alpha/2$ rounded down, and the index on the upper bound is $q_2 = N - q_1 + 1$. These bounds were derived by finding the indices on the sorted means at lower and upper ends of the pdf estimate at which the areas under the pdf estimate equal α. This does not require that the estimated pdf is symmetric.

An example of a practical application in which bootstrap confidence intervals are used can be found in [3].

Chapter 3

Density Estimation

3.1 The Histogram

The histogram of a random variable r is a very simple estimate of a probability density function $f(r)$, based upon measured samples (exemplars) from an experiment. The range of the random variable r is divided (quantized) into bins. The estimate $\widehat{f}(r)$ of the pdf is constructed by counting the number of occurrences of the random variable that lie within each bin. A histogram plot of $\widehat{f}(r)$ is constructed in which the abscissa is the bin number and the ordinate is the number of occurrences of the random variable in each bin. The reader is referred to [64, 39, 42] for a thorough discussion of histograms.

A key issue in the use of histograms is how to choose the number of bins to use. A practical rule of thumb has been developed by Sturges [37] using a combination of theory and empirical studies. This is known as Sturges' Rule for Constructing Histograms. If we let N_E equal the number of exemplars in the data set and N_r equal the number of bins to use for random variable r, then the rule of thumb is

$$N_r = log_2[N_E] + 1 \tag{3.1}$$

This rule of thumb has been found to be useful by the author. Some people argue that Sturges' rule of thumb results in overly-smooth histograms, and Hyndman [38] actually argues that the basic derivation of the rule is incorrect. In practice, other considerations such as ease of visualizing the histogram by visual inspection may be the overriding consideration in choosing the bin size.

3.2 Parzen Kernel pdf Estimation

We also use a more sophisticated pdf estimator of the kernel type [64]. We choose the Parzen kernel type estimator because of its generality, ease of use, and robustness as demonstrated by a wide variety of application observed by the author [64, 65, 67, 68, 11, 12, 13, 14, 15, 16, 17, 18, 19]. The Parzen kernel pdf estimator is the basis for the Probabilistic Neural Network (PNN) proposed by Donald F. Specht [65, 67, 68]. The PNN is actually a Bayes optimal classifier which uses pdf estimates based upon the Parzen kernel. The most commonly used kernel is a Gaussian-shaped kernel.

The essence of the Parzen kernel pdf estimator can be summarized as follows. Let us define the following symbols:

$i =$ Training pattern or exemplar index, where $i = 1, 2, \ldots, m$

$m =$ Number of training or exemplar patterns in the training set

$\underline{X}$ = Pattern or feature vector under test, $\underline{X} = [x_1, x_2, \ldots, x_m]^T$

$\underline{X}_i$ = i-th training vector in the training set

σ = Smoothing parameter for the pdf estimator (to be chosen by the user)

p = Dimension of the feature vector $\underline{X}$ (The dimension of $\underline{X}$ is $p \times 1$)

Given a set of training vectors $\underline{X}_i$ from the training set, the Parzen kernel pdf estimate is given by:

$$\widehat{f}(\underline{X}) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p} \sum_{i=1}^{m} exp\left[-\frac{(\underline{X}-\underline{X}_i)^T(\underline{X}-\underline{X}_i)}{2\sigma^2}\right] \tag{3.2}$$

The kernel for this estimator is a Gaussian-shaped function centered at the exemplar value $\underline{X}_i$ in multi-dimensional space. The width of the Gaussian-shaped kernel is specified by the smoothing parameter σ, which clearly is the standard deviation of the Gaussan kernel. The pdf estimate is the superposition, or summation of all of these kernels summed over the training data set consisting of m samples.

The smoothing parameter choice is very important to the quality of the pdf estimate. As $\sigma \to 0$, the pdf estimate becomes the same as the Nearest Neighbor estimate. Also, the estimated pdf has distinct modes corresponding to the locations of the training samples. As $\sigma \to \infty$ the pdf estimate experiences broad smoothing and interpolation. In addition, the estimated pdf approaches Gaussian and the PNN equals the hyperplane (linear) classifier.

3.3 Creation of the Evenly-Spaced Grid of Feature Values Over which to Compute the pdf Estimate

In practice, we wish to capture estimates of the extreme tails of the density function. Of course, we cannot do this perfectly, so we must engage in the "art" of pdf estimation. The author uses the following technique.

First: for each feature $x_m,\ m = 1, 2, ..., M$, where M is the number of features in the feature vector $\underline{X}$, search for the maximum and minimum values x_m^{Min} and x_m^{Max}, to establish the range of the feature's values $Range(x_m) = [x_m^{Min}, x_m^{Max}]$.

Second: Create the bounds on the evenly-spaced grid over which to calculate the pdf estimate. Do this by expanding the range beyond the minimum and maximum values to be sure that we capture most of the tails of the distribution. A simple way to do this is to choose a scale factor $K_m,\ m = 1, 2, ..., M$ such that $K_m \geqslant 1$. We then create the bounds for the grid for the mth feature as follows:

$$Bounds(x_m) = K_m * Range(x_m) \tag{3.3}$$

Third: Create the Grid for feature x_m by constructing a uniformly-spaced vector of values between the bounds. The user must experiment with the value of K_m to obtain a satisfactory estimate of the tails. Visual inspection of a plot of the pdf estimate is a reasonable way to check the estimate of the tails. The author usually finds that 1.5 is a good starting estimate for the value of K_m.

3.4 Normalization of the pdf Estimate

We know that the integral over a pdf must equal one. In order to ensure this, we sum the values of the pdf estimate to obtain a content S. We then normalize the pdf estimate by dividing it by S.

3.5 Automatic Selection of the Smoothing Parameter

Smoothing parameter selection is generally an ad hoc, manual, empirical, time-consuming activity [64]. We are, therefore interested in automating the process. The MATLAB software written for this project allows the user to choose whether she/he wishes to choose the parameter manually, or automatically. We implemented an automatic scheme suggested by J. B. Cain [66]. This scheme provides a reasonable estimate of the smoothing parameter, given the training data set. If one wishes to adjust σ manually, one can use the Cain algorithm as an initial condition for starting the manual search.

Cain's algorithm is based upon the observation that the pdf estimate at a point (a member of the training set, or set of exemplar samples), should be significantly influenced by more than one exemplar, but not by a large number of exemplars. It is clear that the larger the number of exemplars and the denser the exemplars, the smaller the smoothing parameter σ must be for best performance in estimating the pdf. In the Cain algorithm, σ is set to a constant times the average distance between exemplars in the same class.

Let i denote the exemplar index, and k denote the class index (for our case, $k = 1$). Let ρ_i denote the $i - th$ exemplar from the training set. Let C_k denote the $k - th$ class, and $|C_k|$ denote the number of exemplars in the $k - th$ class.

Let d_i denote the distance between exemplar pattern ρ_i and the nearest exemplar in the class C_k. We can then define the minimum distance between exemplar patterns in class C_k as follows:

$$\widehat{d}_{avg}[k] = \frac{1}{|C_k|} \sum_{\rho_i \in C_k} d_i \tag{3.4}$$

Finally, we assign the smoothing parameter for class C_k to be:

$$\sigma_k = g \bullet \widehat{d}_{avg}[k] \tag{3.5}$$

where $1.1 \leq g \leq 1.4$. The range of g was determined empirically by Cain, and the author has also found it to be useful in a variety of applications [11, 12, 13, 14, 15, 16, 17, 18, 19].

The Cain algorithm is a two-pass algorithm and is summarized as follows:

3.5.1 First Pass

The first pass is nearly identical to the training method used for the PNN:
(1) Present all training patterns (exemplars) ρ_i to the the Parzen estimator.
(2) After all exemplars are presented, set constant the number of exemplars $|C_k|$ in each of the k classes.

3.5.2 Second Pass

Assign the smoothing parameter for class C_k to be [66]:

$$\sigma_k = g \bullet \widehat{d}_{avg}[k] \tag{3.6}$$

where $1.1 \leq g \leq 1.4$, and $\widehat{d}_{avg}[k]$ is given by Equation (3.4).

3.5.3 Other Algorithms for Selecting the Smoothing Parameter

Various researchers have proposed algorithms for automatically choosing the smoothing parameter in a kernel density estimator [64, 69, 70, 71, 72]. This author implemented an algorithm by Silverman [64] for use with this work. It generally serves as good initial estimate of the smoothing parameter.

The formula the author used is given on pages 86-87 of Silverman [64]. Assuming we are given normally-distributed multi-dimensional data with *unit variance*, let us define the following. Let n denote the number of independent, identically distributed (i.i.d.) observations (feature vectors) we have available. Let d denote the length of each feature vector (number of features in a feature vector, or dimensionality). Define the constant $A(K)$, where K refers to the kernel function:

$$A(K) \quad = \{\tfrac{4}{d+2}\}^{1/(d+4)} \qquad \text{(Constant)} \tag{3.7}$$

Now, the optimal window width (or smoothing parameter) for normally-distributed multi-dimensional data with unit variance is given by:

$$h_{opt} = A(K)n^{-1/(d+4)} \tag{3.8}$$

If the data are general (not necessarily Gaussian) with covariance S, if the kernel is radially symmetric and if the data are not transformed (not "not pre-whitened " [64]), then Silverman prescribes the following. Define a single scale parameter, call it σ, and use it to scale the smoothing parameter above to give the smoothing parameter h^*_{opt} as follows:

$$h^*_{opt} = \sigma h_{opt} \tag{3.9}$$

Silverman recommends that a possible choice for σ is the average marginal variance given by

$$\sigma^2 = \frac{1}{d}\sum_i s_{ii} \tag{3.10}$$

This is the formula used in the author's code. Again, this formula may not give the best possible estimate for the smoothing parameter, but it is useful as an initial guess. Please see the next section for additional ideas for searching for the optimal smoothing parameter in a practical application.

Chapter 4

Bayesian Classification Using the Probabilistic Neural Network

The algorithms developed in earlier chapters are used in practical applications of Bayes detection theory and classification theory. This chapter summarizes the practical aspects of applying the theory.

We assume throughout this discussion that the cost of an incorrect decision is higher than the cost of a correct decision. In other words, $C_{10} > C_{00}$ and $C_{01} > C_{11}$. Under this assumption, the detector that minimizes the Bayes risk is given by the following:

$$\frac{f(\underline{X}|H_1)}{f(\underline{X}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P(H_0)(C_{10}-C_{00})}{P(H_1)(C_{01}-C_{11})} \qquad \text{(Bayes Decision Criterion)} \tag{4.1}$$

The ratio of the conditional densities is called the likelihood ratio and is denoted by $\Lambda(\underline{X})$:

$$\Lambda(\underline{X}) = \frac{f(\underline{X}|H_1)}{f(\underline{X}|H_0)} \qquad \text{(Likelihood Ratio)} \tag{4.2}$$

The quantity on the right hand side of the relation (4.1) is the threshold of the test and is denoted by η:

$$\eta \triangleq \frac{P(H_0)(C_{10}-C_{00})}{P(H_1)(C_{01}-C_{11})} \qquad \text{(Threshold)} \tag{4.3}$$

Thus, the Bayes criterion leads to a likelihood ratio test:

$$\Lambda(\underline{X}) \underset{H_0}{\overset{H_1}{\gtrless}} \eta \qquad \text{(Likelihood Ratio Test)} \tag{4.4}$$

We see that the test threshold allows for weighting according to the priors and the costs. This allows the user flexibility in choosing a threshold that is best for the problem at hand.

Another equivalent way to view the Bayes likelihood ratio test is to rewrite it in terms of conditional posterior probabilities. Under the assumptions above, the conditional posterior $P(H_1|\underline{X})$ for the binary hypothesis case can be written as follows [82].

$$P(H_1|\underline{X}) = \frac{f(\underline{X}|H_1)P(H_1)}{f(\underline{X}|H_0)P(H_0) + f(\underline{X}|H_1)P(H_1)} \qquad \text{(Posterior Probability)} \tag{4.5}$$

Similarly, we can write the conditional posterior $P(H_0|\underline{X})$ for the binary hypothesis case can be written as follows [82]:

$$P(H_0|\underline{X}) = \frac{f(\underline{X}|H_0)P(H_0)}{f(\underline{X}|H_0)P(H_0) + f(\underline{X}|H_1)P(H_1)} \qquad \text{(Posterior Probability)} \tag{4.6}$$

One can show that the Bayes decision rule can take the following equivalent form:

$$P(H_1|\underline{X}) \underset{H_0}{\overset{H_1}{\gtrless}} .5 \qquad \text{(Posterior Probability Test)} \tag{4.7}$$

Examples of practical applications in which this formulation are applied appear in [12, 11].

A comparison of the PNN with other classifiers such as the back propagation neural network [50] is summarized in [12]. Briefly, the PNN offers advantages in training time (it learns in just one pass through the data), it lacks local minima, it has better generalization capability, and it has better classification performance when the data are sparse. Its main drawback is that it requires that the training data must be stored, but this problem can be mitigated.

4.1 Practical Aspects of Choosing the Smoothing Parameter

We use two main methods for choosing the smoothing parameter, σ; manually and automatically. (1) In the manual mode, we simply choose values of σ and compute the resulting probability of correct classification P(CC). The curve for P(CC) generally has a knee or maximum, and we choose the value of that maximizes P(CC). (2) In the automatic mode, we build a loop into the PNN software that allows us to try automatically a range of values for σ and map out a curve for P(CC) vs. σ , which we can automatically search for the maximum. One possible disadvantage of this technique is that the same value of σ is used for both classes, H_1 and H_0. In general, we have the flexibility to use a distinct value σ_{H_1} for class H_1, and σ_{H_0} for class H_0. This requires a search in the 3D space of P(CC) vs. σ_{H_1} vs. σ_{H_0}. If one is willing to pay the price of additional computational complexity, an "adaptive PNN "scheme can sometimes be beneficial [65, 68, 66].

Specht [65] reports that the P(CC) vs. σ curve has a broad maximum, so the performance of the pdf estimator is relatively insensitive to the exact value of σ chosen. In fact, for a mine detection problem [12], we have found that this rule of thumb is valid. However, for another application with a small sample size, we found that relatively small changes in the value of the smoothing parameter significantly altered the classification results [13, 18, 19, 17]. Therefore, for those data, it is very important to find the optimum σ , and this becomes an important reason why we chose to use an automatic tuning algorithm..

Chapter 5

Feature Selection

5.1 Lower Bound on the Number of Independent Training Samples Necessary for Classification

We must pay careful attention to an important relationship between the number of features used and the sample size. A combination of theoretical and empirical studies has led to the following rule of thumb [47, 52]. Given that we have chosen the number of features in our feature vector, then:

$$\text{Number of Independent Training Samples Needed Per Class} \geqslant (5)\ \text{(Number of Features)} \tag{5.1}$$

For example, if we have a feature vector of dimension 10, then we need at least 50 training samples in each class to support the classifier. Many researchers recommend using many more than five times the dimension of the feature vector. This rule of thumb has proved to be of value in mine detection and a variety of other applications the authors have studied [50]. The theoretical reasoning for the rule of thumb is based upon the fact that covariance matrices are used in feature space class separability measures and in many classification algorithms. The rule of thumb reflects the number of training samples required to ensure in practice that the covariance matrix is estimated with sufficient precision [47, 52].

5.2 Upper Bound on the Number of Features One Can Use

An important implication of this rule of thumb is an upper bound on the number of features to use, given the number of independent training samples. So, given that we have a limited number of training samples, then:

$$\text{Number of Features} \leqslant \left(\frac{1}{5}\right)\ \text{(Number of Independent Training Samples Per Class)} \tag{5.2}$$

Note that if the sample size is small, it severely limits the number of features we can use. For example, if we have only 15 training samples available per class, then we can use at most about 3 features.

5.3 Measures of Class Separability

The criterion we use to judge the feature set is a measure of class separability consisting of a probabilistic measure of the distance between classes in the corresponding feature space. In general, the complete information about

the probabilistic structure of the classes can be given in terms of the class conditional probability density function $f(X|Class_i)$ and the class prior probabilities P_i, where X is the feature vector, $i = 1, 2, ...N$ is the class index and N is the number of classes. Any function $C(\cdot)$

$$C(\cdot) = \int g\Big[f(X|Class_i), f(X|Class_j), P_i, P_j\Big]dX \tag{5.3}$$

satisfying the following properties can be used as the distance measure: (1) $C(\cdot)$ is nonnegative. (2) The maximum value of $C(\cdot)$ occurs when the classes are disjoint in the feature set, and (3) The distance $C(\cdot)$ equals zero when the class conditional probability density functions are identical [43, 42], can be used as a measure of class separability. $g[\cdot]$ is the criterion function determining the probabilistic dependence between patterns and classes.

The Bhattacharyya distance $J(i, j)$, is defined by Devijver and Kittler as follows [70, 42]:

$$J(i,j) = \frac{1}{8}\Big(\underline{\mu}_j - \underline{\mu}_i\Big)^T\left[\frac{\Sigma_i + \Sigma_j}{2}\right]^{-1}\Big(\underline{\mu}_j - \underline{\mu}_i\Big) + \frac{1}{2}ln\frac{\Big|\frac{1}{2}\Big(\Sigma_i + \Sigma_j\Big)\Big|}{\Big[|\Sigma_i||\Sigma_j|\Big]^{\frac{1}{2}}} \tag{5.4}$$

where $\underline{\mu}_i$ and $\underline{\mu}_j$ are the mean vectors computed over the feature vectors in classes i and j, and Σ_i and Σ_j are the corresponding covariance matrices for classes i and j. If we assume that the two class covariance matrices are equal, $\Sigma_i = \Sigma_j = \Sigma$, then we assume that a mean covariance matrix over all the classes is sufficient to characterize the feature space. If we then apply this assumption to the Bhattacharyya distance, we obtain the Mahalanobis distance 4,25:

$$J(i,j) = \frac{1}{8}\Big(\underline{\mu}_j - \underline{\mu}_i\Big)^T\Sigma^{-1}\Big(\underline{\mu}_j - \underline{\mu}_i\Big) \tag{5.5}$$

We construct the overall measure C discussed above by approximating the integrals in Equation (5.3) with summations to form the following:

$$C = \sum_{i=1}^{N}\sum_{j=i+1}^{N} J(i,j)P_iP_j \tag{5.6}$$

In many practical scenarios, it is decided that the samples available are not sufficient to justify the estimation of priors. Therefore, having no reliable knowledge of the prior probabilities for the classes, we can assume they are equal, and we obtain Equation (5.7).

$$C = \sum_{i=1}^{N}\sum_{j=i+1}^{N} J(i,j) \tag{5.7}$$

5.4 The Number of Possible Feature Sets

In choosing a feature set, we could simply use the brute force method of conducting an exhaustive search of all possible combinations of features and choose the one with best value of a performance criterion. If we were to

choose b features from a much larger set of B features, this would involve the evaluation of B features taken b at a time, or

$$\text{No. of Possible Feature Subsets} = \frac{B!}{b!(B-b)!} \tag{5.8}$$

subsets. This number can be very large, even for a relatively small number of features. For example, selecting 10 features from a set of 20 involves 184,756 evaluations. This exhaustive search method is very undesirable because of this computational complexity, so other more efficient methods are used.

5.5 The Branch and Bound Algorithm

We discuss the Branch and Bound Algorithm for feature selection [20, **?**] because it is a standard of comparison for other algorithms. It is globally optimal in the sense that it finds the optimal feature set, but it generally does not require as much computation time as the exhaustive search method. The worst case scenario using Branch and Bound can be as computationally burdensome as the exhaustive search, but that case is very rare. Branch and Bound avoids exhaustive search by rejecting suboptimal subsets without direct evaluation and guarantees that the selected subset yields the globally best value of any criterion function J that satisfies a monotonicity condition:

$$J_1(x_1) \geqslant J_2(x_1, x_2) \geqslant \ldots J_m(x_1, x_2, \ldots, x_m) \tag{5.9}$$

where $J_i(x_1, x_2, \ldots, x_i)$ is the criterion function evaluated for all features *except* $x_1, x_2, \ldots, x_i$ from the feature set. The restriction of monotonicity is not severe and is not a limitation in practice, because it simply requires that a set of S features is at least as good as any proper subset of itself for the purpose of class separation. A large number of criteria, including the Bhattacharyya criterion satisfy the monotonicity condition [20, 21, 26]. The key point is that even though the Branch and Bound algorithm finds the optimal feature set, and even though it is much more efficient than the exhaustive search, it nonetheless usually involves a very large number of computations, making it often undesirable in practice. For many applications, we have a large feature set to search, so in order to achieve acceptable computational complexity, we are willing to accept an algorithm that produces a suboptimal feature set. We often prefer to use such an algorithm, the Sequential Forward Selection Algorithm, which we describe next.

5.6 The Sequential Forward Selection (SFS) Algorithm

Using the measure of class separability, we can rank order the features in the feature set according to their contribution to class separability. To achieve this, we use the Sequential Forward Selection (SFS) Algorithm [20]. SFS uses a bottom-up search strategy, in which we start with the null set of features. One feature at a time is included in the current feature set. At each iteration, the feature to be included in the feature set is selected from among the remaining available features, so that the enlarged set of features yields a maximum value of the criterion function used. Note that there is a corresponding Sequential Backward Selection algorithm, which starts with a full set of features and eliminates them one-by-one until it reaches the desired number of features [20].

For SFS, we first specify a priori the number of features b desired in the final feature set. At each stage in the algorithm, one feature is added to the current feature set. The new feature is selected from the set of features not already in the current feature set. For a new feature to be included, the new enlarged feature set must yield the maximum value of C. The algorithm is described as follows:

Let x_i be an individual feature, which is an element of the set of all B features under consideration, $\underline{X} =$

$\{x_i, i = 1, 2, \ldots, B-1\}$. So, we can write the overall feature vector as:

$$\underline{X} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{B-1} \end{bmatrix} \tag{5.10}$$

Let $\underline{X}_k$ be the selected feature set at algorithm iteration index k. Initialize the algorithm by setting the feature set $\underline{X}_0$ to the null set and specifying the desired number of features b in the final feature set. At the kth step, k features have been selected from $\underline{X}$ to form the feature set $\underline{X}_k$. Let $\underline{X} - \underline{X}_k$ be the set of available (not yet selected) features. Rank the elements a_i of the set $\underline{X} - \underline{X}_k$ so that

$$C(\underline{X}_k \cup \{a_1\}) \geqslant C(\underline{X}_k \cup \{a_2\}) \geqslant C(\underline{X}_k \cup \{a_{(B-k)}\}) \tag{5.11}$$

Thus, at the (k+1)st step, we choose $\underline{X}_{k+1} = \underline{X}_k \cup \{a_1\}$, because it results in the largest separability measure. Continue to include new features in the feature set until the required number of features b is selected.

Use of the SFS algorithm involves some important tradeoffs. The main *positive characteristics* of the SFS algorithm are that it is simple, easy to implement, has a small computational complexity and performs well. It runs rapidly and has produced excellent selection results for a wide variety of real data sets and applications, including the one described in this paper [20, 12, 13, 18, 17, 19].

The main *drawback* of the SFS algorithm is that it does not test all possible combinations of feature sets. Notice that once a feature has been included in the set, it cannot be removed. In general, this could lead to undesirable results, because a feature included at step k could become irrelevant or redundant due to the effects of features included at subsequent steps. For this reason, the SFS algorithm is suboptimal in the sense that the selected feature set is not necessarily the one that would be selected if all possible combinations of feature sets were tested. For many applications in the past, the author has found the performance of the SFS algorithm to be generally satisfactory, and the optimality of the Branch and Bound algorithm was not worth the high computational cost. We often have a very large feature set to search, and the feature sets found by SFS provided excellent classification performance.

Table 1: The Sequential Forward Selection Algorithm

GIVEN: A set $\underline{X}$ containing B features from which it is desired to select a subset of b features ($b \leqslant B$).
Let $\underline{X}_k$ denote the selected feature set at algorithm iteration k.

1. Initialize the algorithm by setting the feature set $\underline{X}_0 = \{\}$ (the null set)

2. Specify b, the number of features desired in the final selected feature set.

3. kth Iteration ($k = 1, 2, 3, \ldots, b-1$)

Rank order the elements a_i of the set $\underline{X} - \underline{X}_k$ so that

$$C(\underline{X}_k \cup \{a_1\}) \geqslant C(\underline{X}_k \cup \{a_2\}) \geqslant C(\underline{X}_k \cup \{a_{(B-k)}\}) \tag{5.12}$$

Let $\underline{X}_{k+1} = \underline{X}_k \cup \{a1\}$

4. Repeat step 3 until $k + 1 = b$, then stop. The final feature set is $\underline{X}_b$.

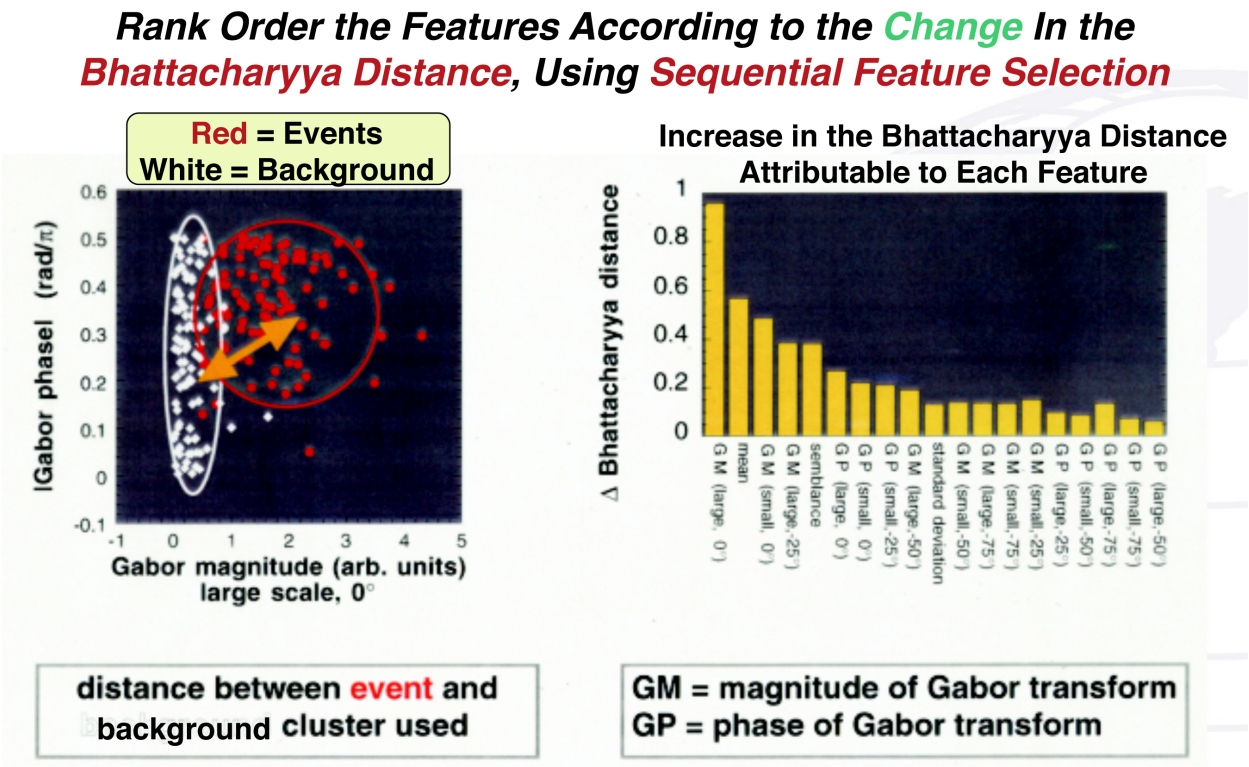


Figure 5.1: Feature selection results using the Bhattacharyya distance and the Sequential Forward Selection algorithm are shown in this figure. The application is automatic event detection for seismic oil exploration performed for Shell Oil [11, 16]. A large set of features included mean, standard deviation, Gabor transform magnitude, Gabor transform phase and others. The figure on the left depicts a two-dimensional slice through feature space for two of the Gabor transform features. The figure on the right depicts the change in the Bhattacharyya distance vs. feature index (or name). One can see that beyond about nine or ten features, the value added by each feature is small. Hence, we used a subset of about nine features.

5.7 Feature Selection Results for the PAT Data Set

This unclassified example is from a project dubbed "PAT, " and the data set was dubbed "k35."

A small data set was created as a subset of the full k35 original data set. See Figure (5.2) for a description of the data set parameters. Figure (5.3) shows the the Sequential Forward Selection (SFS) results using the Mahalanobis distance. Figure (5.4) shows the the SFS results using the Bhattacharyya distance. Note that both distance criteria gave the same feature ranking for this data set. In these figures, the features are not ranked in order of decreasing change in the distance measure.

Figure (5.5 and Figure (5.6 depict the same results corresponding to Figure (5.3) and Figure (5.4), except that the features are ranked in order of decreasing change in the distance measure.

Another result helped give confidence in the algorithm. When the SFS algorithm was asked to rank various numbers of features ($b = 1, 2, \ldots, 6$) , the ranking order produced was the same as that shown for the case in which all 6 of 6 features were selected (2 1 6 3 4 5). One cannot expect such results in general.

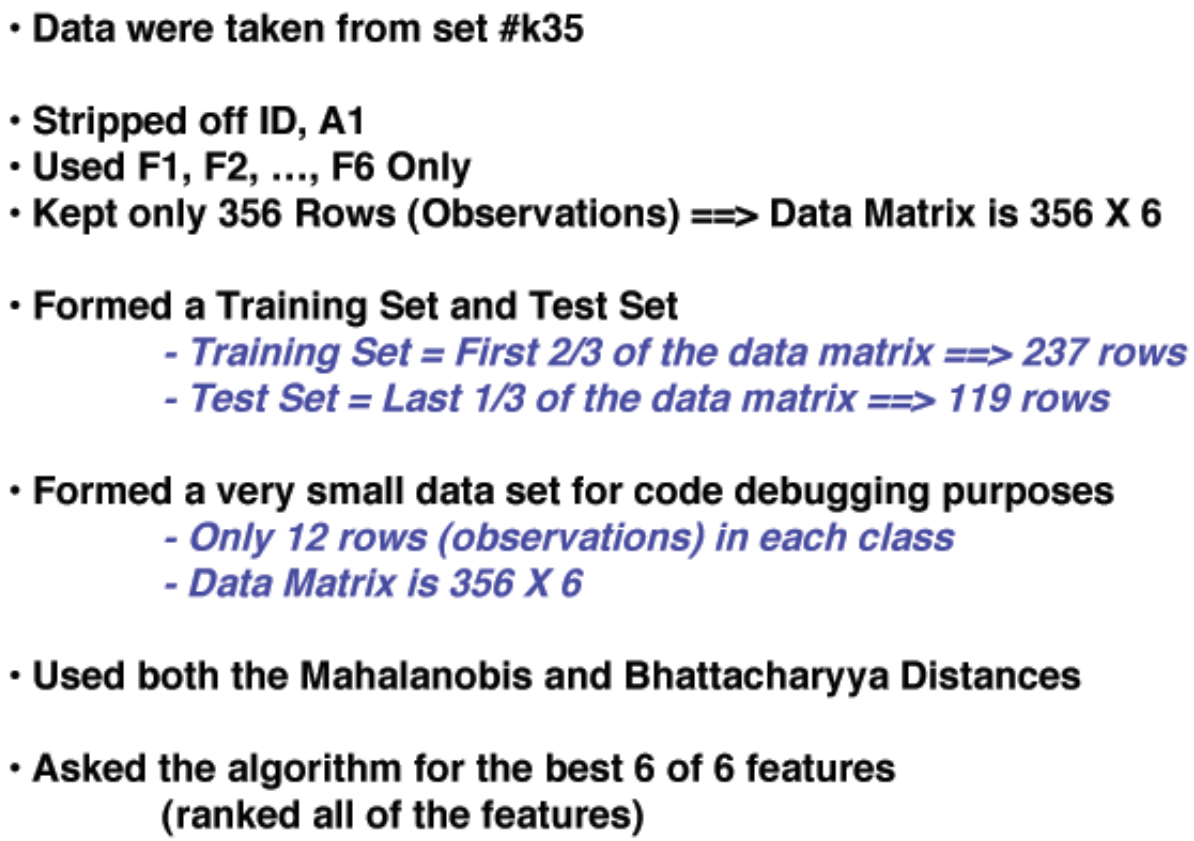


Figure 5.2: A small subset of the original k35 data set was used for the examples in this section.

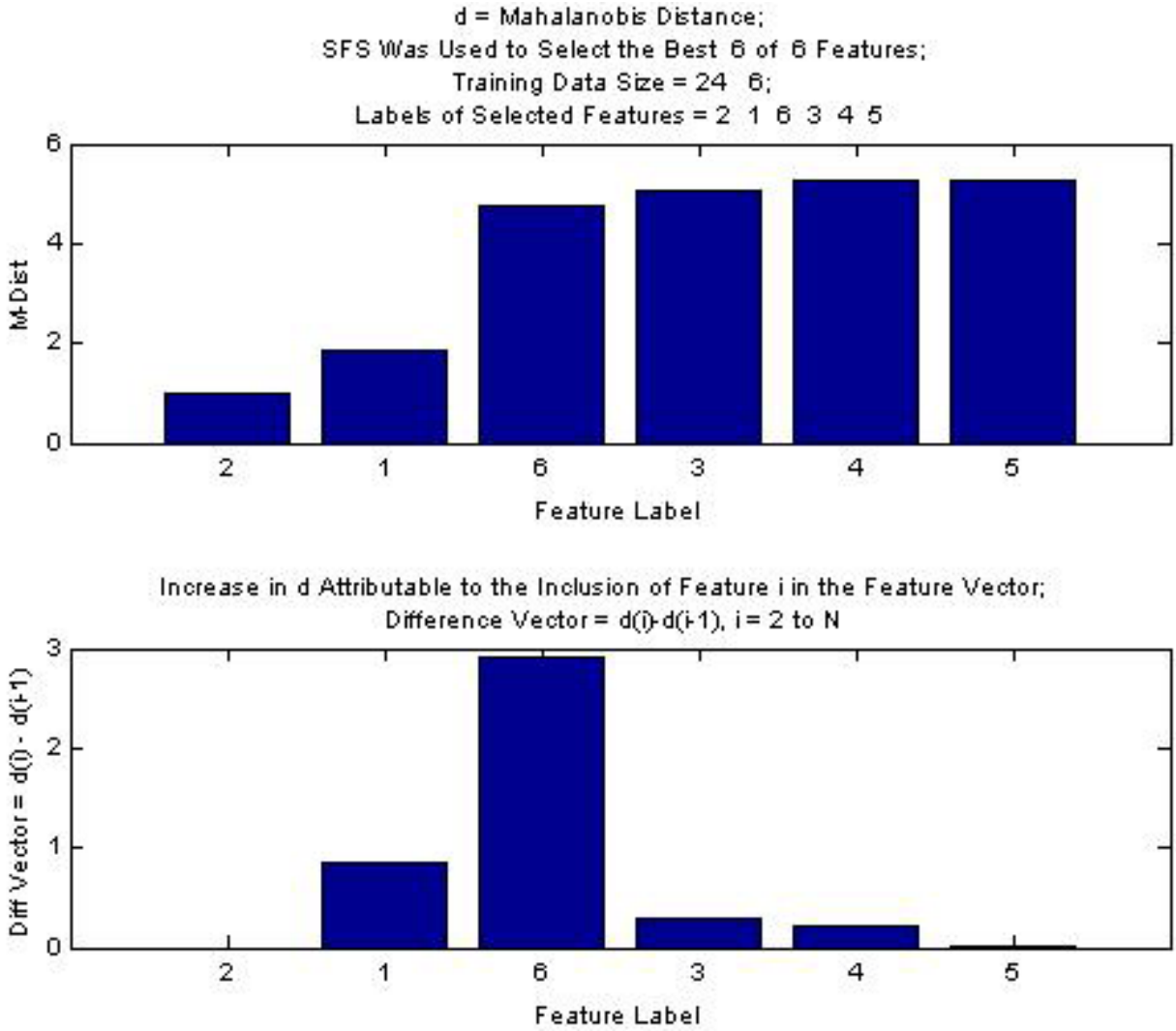


Figure 5.3: Feature Selection Results for the PAT Data Set: (Top Figure) Mahalanobis Distance plotted versus the labels of the features selected in rank order, (Bottom Figure) Increase in the Bhattacharyya Distance attributable to the inclusion of Feature i in the feature vector. The differences in this figure are not rank-ordered.

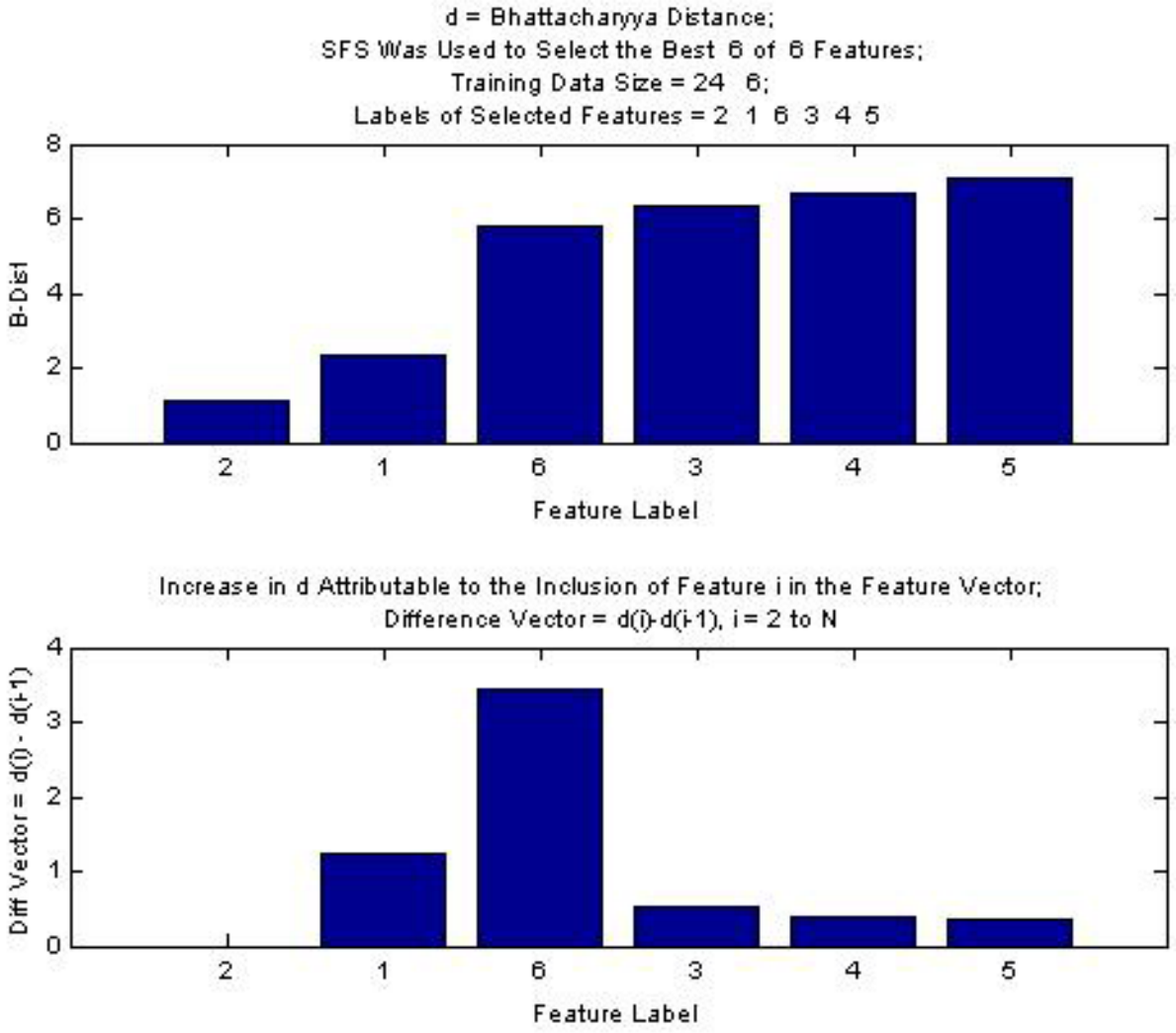


Figure 5.4: Feature Selection Results for the PAT Data Set: (Top Figure) Bhattacharyya Distance plotted versus the labels of the features selected in rank order, (Bottom Figure) Increase in the Bhattacharyya Distance attributable to the inclusion of Feature i in the feature vector. The differences in this figure are not rank-ordered.

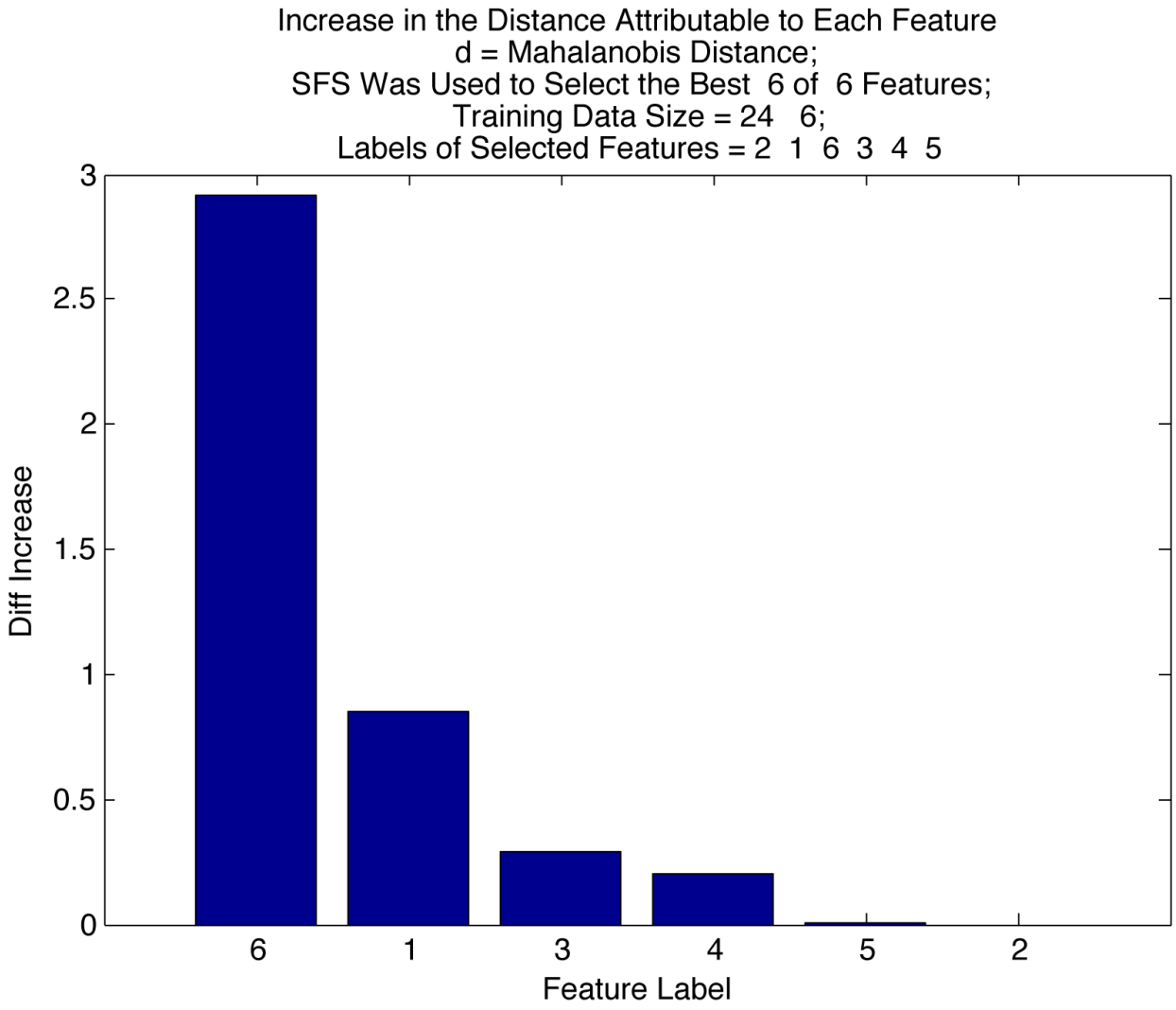


Figure 5.5: Feature Selection Results for the PAT Data Set: Increase in the Difference of the Mahalanobis Distance plotted versus the labels of the features selected in rank order. This display makes it easy for the user to see the value added by each feature. The user can then easily choose a subset of the features to use. For example, in this figure, one might argue that using more than four features adds very little to the distance measure; so one might choose to use only the first four of the six features.

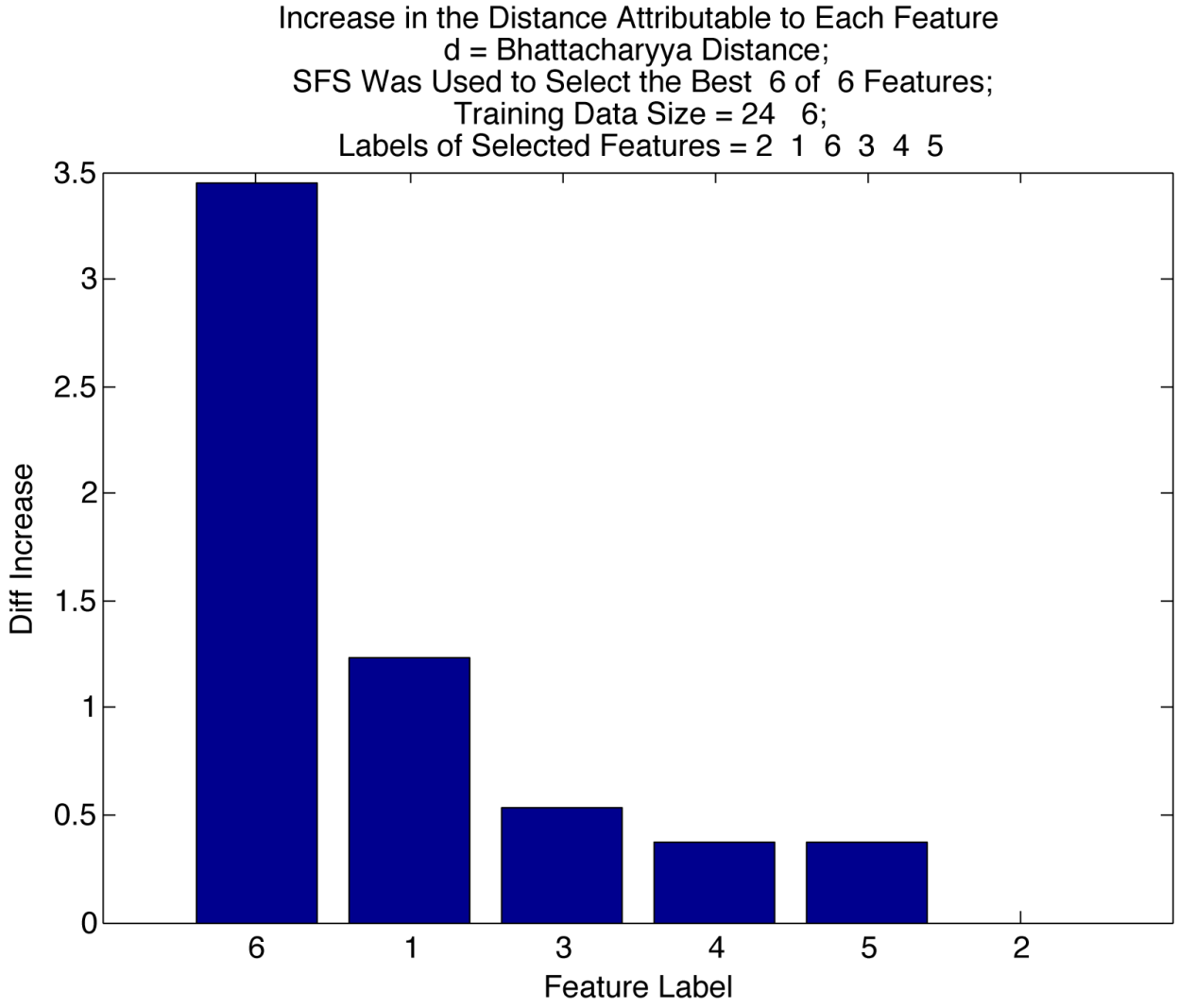


Figure 5.6: Feature Selection Results for the PAT Data Set: Increase in the Difference of the Bhattacharyya Distance plotted versus the labels of the features selected in rank order. This display makes it easy for the user to see the value added by each feature. The user can then easily choose a subset of the features to use. For example, in this figure, one might argue that using more than five features adds very little to the distance measure; so one might choose to use only the first five of the six features. Or, the user might decide to use only the first three features, because the increase in the distance measure is small after three.

Chapter 6

Statistical/ Information Theoretic Distance Metrics and the Hellinger Distance

6.1 Desired Properties of a Distance Measure

For a distance measure to be useful in measuring the separation of probability densities, it is desirable that the distance possess the four properties of a metric in the mathematical sense. If we let $d[f(x), g(x)]$ denote the distance between two pdf's $f(x)$ and $g(x)$, then the four properties of a metric are [27, 28, 29]:

(1) **Identity:** $d[f(x), g(x)] = 0$ if $f(x) = g(x)$
The distance between two like objects should be minimum.

(2) **Non-Negativity:** $d[f(x), g(x)] \leq 0$
To conform with traditional concepts of distance, the distance should be non-negative. This implies that the distance from an object to itself should be zero, and the distance between two objects should be zero only if the two objects are the same.

(3) **Symmetry:** $d[f(x), g(x)] = d[g(x), f(x)]$
A desirable property is symmetry, meaning that the distance between $f(x)$ and $g(x)$ is the same as the distance between $g(x)$ and $f(x)$.

(4) **Triangle Inequality:** $d[f(x), h(x)] \leq d[f(x), g(x)] + d[g(x), h(x)]$
A distance measure should obey the triangle inequality; that is, the distance between $f(x)$ and $g(x)$ plus the distance between $g(x)$ and $h(x)$ should be less than or equal to the distance between $f(x)$ and $h(x)$. This allows the distances among objects to be compared easily and reinforces the traditional concept of distance.

6.2 Divergence

We are interested in defining a reasonable metric with which we can measure the separation between two probability density functions $f(x)$ and $g(x)$. The statistical and information theoretic literature offers the concept of divergence and a variety of ways to define it [26, 25, 27, 28, 29]. Various divergence measures are defined in terms of the likelihood ratio of the two densities:

$$\lambda(x) = \frac{g(x)}{f(x)} \tag{6.1}$$

A classic divergence is the ϕ-divergence defined as [26, 25, 27, 28, 29]:

$$d_\phi(f,g) = E_{F(x)}[\phi(\lambda(x))] = \int_X \phi(\lambda(x))dF(x) = \int_X \phi(\frac{g(x)}{f(x)})f(x)dx \tag{6.2}$$

where $F(x)$ is a cumulative distribution function, $f(x) = \frac{dF(x)}{dx}$ and $g(x)$ are probability density functions, and $\phi(\lambda)$ is a convex function for $\lambda > 0$ with $\phi(1) = 0$.

Various definitions of $\phi(\lambda)$ produce various definitions of divergence. We now examine several ϕ-divergences and focus on their suitability for use as metrics and distance measures for feature selection purposes.

6.2.1 The Kullback-Leibler (KL) Divergence

The Kullback-Leibler (KL) Divergence is defined as

$$d_{KL}(f,g) = \int_X g(x)log(\frac{g(x)}{f(x)})dx \tag{6.3}$$

The KL Divergence is also known by other names, including directed divergence, relative entropy, and information divergence [26, 25, 27, 28, **?**] It corresponds to the ϕ-divergence when

$$\phi(\lambda) = \lambda log(\lambda) \tag{6.4}$$

The KL divergence satisfies the Identity and Non-negativity properties, but it is NOT symmetric and does NOT satisfy the triangle inequality. Therefore, it does not satisfy all of the properties of a metric. However, It is often used in communications and message coding applications [26, 25, 27, 28, 29].

The divergence, or relative entropy has been proved to satisfy the monotonicity property we desire for feature selection. The monotonicity property is summarized in Equation (5.9). The proof of monotonicity appears in [26].

6.2.2 The Symmetric Kullback-Leibler Divergence

The KL divergence has been extended to form the Symmetric KL Divergence, defined as the sum of the KL divergence calculated in both directions as follows:

$$d_{KLS}(f,g) = d_{KL}(f,g) + d_{KL}(g,f) \tag{6.5}$$

Some definitions include a factor of one-half as a normalization method, but this is not strictly required. The symmetric KL divergence corresponds to the ϕ-divergence when

$$\phi(\lambda) = (\lambda - 1)log(\lambda) \tag{6.6}$$

This distance satisfies the first three properties of a metric, AND it is symmetric. However, it does not obey the triangle inequality. The applications in which it is used are similar to those of the KL divergence.

6.2.3 The Bhattacharyya Distance

The Bhattacharyya distance is also a ϕ-distance, and is defined as follows:

$$d_B(f,g) = \int_X \sqrt{f(x)g(x)}dx \tag{6.7}$$

This corresponds to the ϕ-divergence when

$$\phi(\lambda) = \sqrt{\lambda} \tag{6.8}$$

The Bhattacharyya distance satisfies the first three properties of a metric, but it does NOT obey the triangle inequality. A very desirable property of the Bhattacharyya distance is that its range is $[0, 1]$, making it very attractive for comparing distances.

6.2.4 The Hellinger Distance

The squared Hellinger distance is defined as follows:

$$d_H^2(f, g) = \frac{1}{2} \int_X [\sqrt{f(x)} - \sqrt{g(x)}\,]^2 dx \tag{6.9}$$

This corresponds to the ϕ-divergence when

$$\phi(\lambda) = \frac{1}{2}(\sqrt{\lambda} - 1)^2 \tag{6.10}$$

Moving the square root to the right-hand side of the equation, the Hellinger distance is given by:

$$d_H(f, g) = \frac{1}{\sqrt{2}} \sqrt{\int_X [\sqrt{f(x)} - \sqrt{g(x)}\,]^2 dx} \tag{6.11}$$

The Hellinger distance satisfies all four properties of a metric and its range is $[0, 1]$. This makes it an ideal candidate for use in estimation, feature selection and classification problems. The robustness of minimum Hellinger distance methods have been explored in [26, 25, 28, 31, 33, 34, 30].

The divergence, or relative entropy has been proved to satisfy the monotonicity property we desire for feature selection. The monotonicity property is summarized in Equation (5.9). The proof of monotonicity appears in [26]. Because the Hellinger distance is a form of divergence (see above), it satisfies the desired monotonicity property.

6.3 The Hellinger Distance Written for Discrete Random Variables in Vector Form

In order to use the Hellinger distance for feature selection, we need to write it in a form that can be programmed on a digital computer. We need to write it for discrete random variables (using summations rather than integrals). We must also write it so that the scalar feature x is replaced by the vector feature vector $\underline{X}$. Thus, we are dealing with multivariate pdf estimates $f(\underline{X})$.

6.3.1 The Scalar Hellinger Distance Written for Discrete Random Variables

For the scalar case, replace the continuous random variable x with the discrete random variable x_k. Let the integer discrete index be defined as $k = 0, 1, \ldots, K-1$, where K is the integer number of measured samples of $x(k)$ available for the computation. Then, replace the integral with the appropriate sum as follows to arrive at the scalar discrete squared Hellinger distance:

$$d_H^2(f, g) = \frac{1}{2} \sum_{k=0}^{K-1} [\sqrt{f(x_k)} - \sqrt{g(x_k)}\,]^2 dx \tag{6.12}$$

6.3.2 The Multivariate (Vector) Hellinger Distance Written for Discrete Random Variables

We now wish to compute the squared Hellinger distance between two multivariate densities $f(\underline{X})$ and $g(\underline{X})$. $\underline{X}$ is an $N \times 1$ feature vector containing N features. We have M measured training feature vectors $\underline{X}_m$, $m = 1, 2, \ldots, M$ that we can use for computing the Hellinger distance, which we now write as:

$$d_H^2(f,g) = \frac{1}{2} \sum_{m=0}^{M-1} [\sqrt{f(\underline{X}_m)} - \sqrt{g(\underline{X}_m)}\,]^2 d\underline{X} \tag{6.13}$$

For the two-dimensional case ($N = 2$), we can define two discrete indices to use for computing the distance above. Let $k_1 = 0, 1, 2, \ldots, N_{x1} - 1$ and $k_2 = 0, 1, 2, \ldots, N_{x2} - 1$. We define the sampling intervals for k_1 and k_2 to be dx_1 and dx_2. Then, we can write the squared Hellinger distance as follows:

$$d_H^2(f,g) = \frac{1}{2} \sum_{k_1=0}^{N_{x_1}-1} \sum_{k_2=0}^{N_{x_2}-1} [\sqrt{f(k_1,k_2)} - \sqrt{g(k_1,k_2)}\,]^2 dx_1 dx_2 \tag{6.14}$$

Of course, the expression for higher-dimensional cases can be written as a simple extension of this expression.

Chapter 7

Example 1: Simulation of 2D Random Variables, 2D pdf Estimation and 2D Hellinger Distance Computation

The purpose of this experiment is to demonstrate the performance of the new multi-dimensional pdf estimation code written for this work. In order to do this, we simulate two sets of data (labeled X0 and X1) for which we wish to estimate pdfs. Finally, after the pdf estimates are evaluated, we compute the Hellinger distance between the two pdfs of the two data sets.

7.1 The X0 Data Set: Bivariate Bimodal Gaussian Distributed

This data set was simulated using a Gaussian random number generator. The number of features in the feature vector is two (it is a bivariate random variable). The number of two-dimensional data vectors (statistical samples) generated is 100. The mean vector $= [0 \;\; 0]^T$ and the covariance matrix is a 2×2 identity matrix $I_{2\times 2}$.

Figure (7.1) depicts a scatter plot of the two-dimensional data set X0 in feature space. The kernel density pdf estimate for set X0 is displayed as a color image in Figure (7.2). The estimator used a Gaussian kernel with smoothing parameter $\sigma = 0.3$. The kernel density pdf estimate for set X0 is displayed as a grey scale image in Figure (7.3). The estimator used a Gaussian kernel with smoothing parameter $\sigma = 0.3$.

7.2 The X1 Data Set: Bivariate Bimodal Gaussian Distributed

This data set was simulated using a Gaussian random number generator. The number of features in the feature vector is two (it is a bivariate random variable). The number of two-dimensional data vectors (statistical samples) generated is 30.

Let us define $\underline{x} \sim N[\mu, \sigma]_{N\times 1}$ as the notation for an $N \times 1$ vector of samples of a scalar random variable x that is Normally distributed with mean μ and standard deviation σ. Using this notation, we define the 30×2 matrix $X1$ that contains the simulated data as shown below.

$$X1 = \begin{bmatrix} \underline{x}_{11} \sim N[0, .5]_{20\times 1} & \underline{x}_{12} \sim N[5, 2.5]_{20\times 1} \\ \underline{x}_{21} \sim N[.75, .25]_{10\times 1} & \underline{x}_{22} \sim N[8.75, 1.25]_{10\times 1} \end{bmatrix} \qquad \text{(Data Matrix)} \qquad (7.1)$$

Figure 7.4 depicts a scatter plot of the two-dimensional data set X1 in feature space. The abscissa corresponds to the first column of the data matrix X, and the ordinate corresponds to the second column of the data matrix X.

The kernel density pdf estimate for set X1 is displayed as a color image in Figure 7.5. The estimator used an Epanechnikov kernel with smoothing parameter $\sigma = 0.3$.

The kernel density pdf estimate for set X1 is displayed as a grey scale image in Figure 7.6. The estimator used an Epanechnikov kernel with smoothing parameter $\sigma = 0.3$.

The kernel density pdf estimate for set X1 is displayed as a three-dimensional plot in Figure 7.7. Note that the circles depict the original observation data samples from which the estimated was calculated.

7.3 Hellinger Distance Results

The calculated Hellinger distance between X0 and X1 for this example is $hD2D = 0.175$.

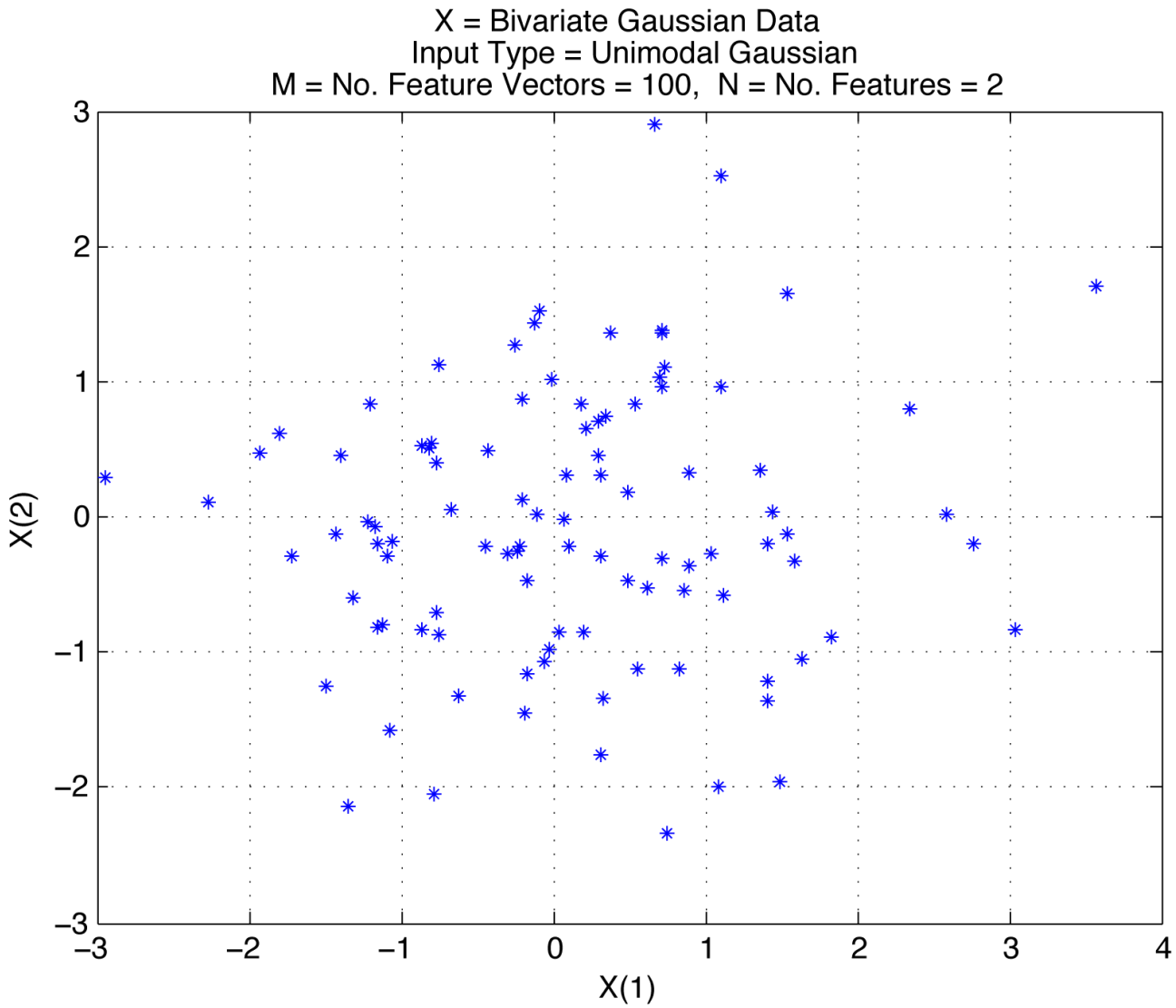


Figure 7.1: Example 1, X0: Scatter plot of simulated observation data for the unimodal bivariate Gaussian-distributed random variable X0. The mean vector $= [0 \;\; 0]^T$ and the covariance matrix is a 2×2 identity matrix $I_{2\times 2}$.

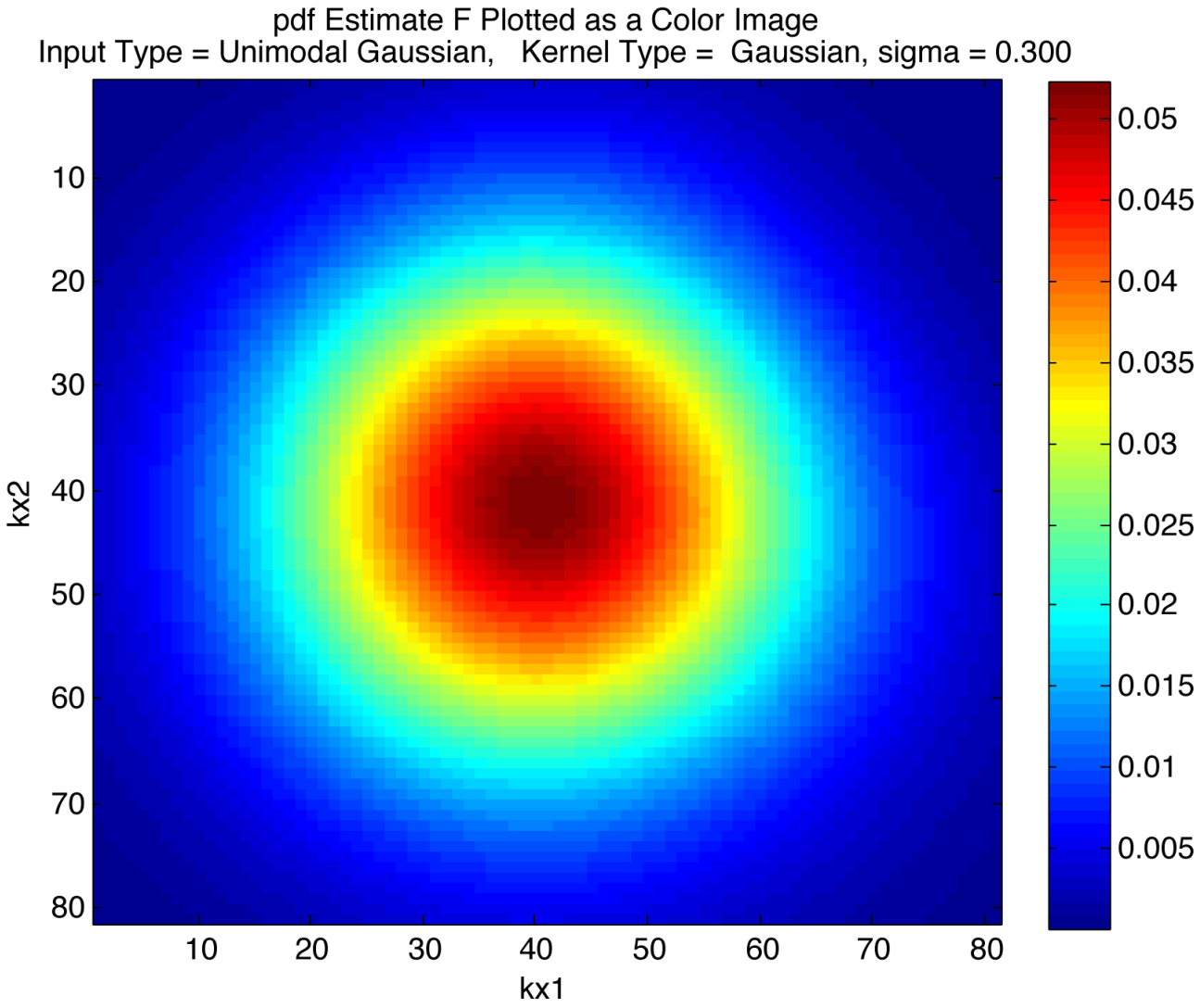


Figure 7.2: Example 1, X0: The kernel density pdf estimate $f(X0)$ for data set X0 is displayed as a color image in this figure. The estimator used a Gaussian kernel with smoothing parameter $\sigma = 0.3$. The 2D grid for the pdf estimate is denoted $(kx1, kx2)$.

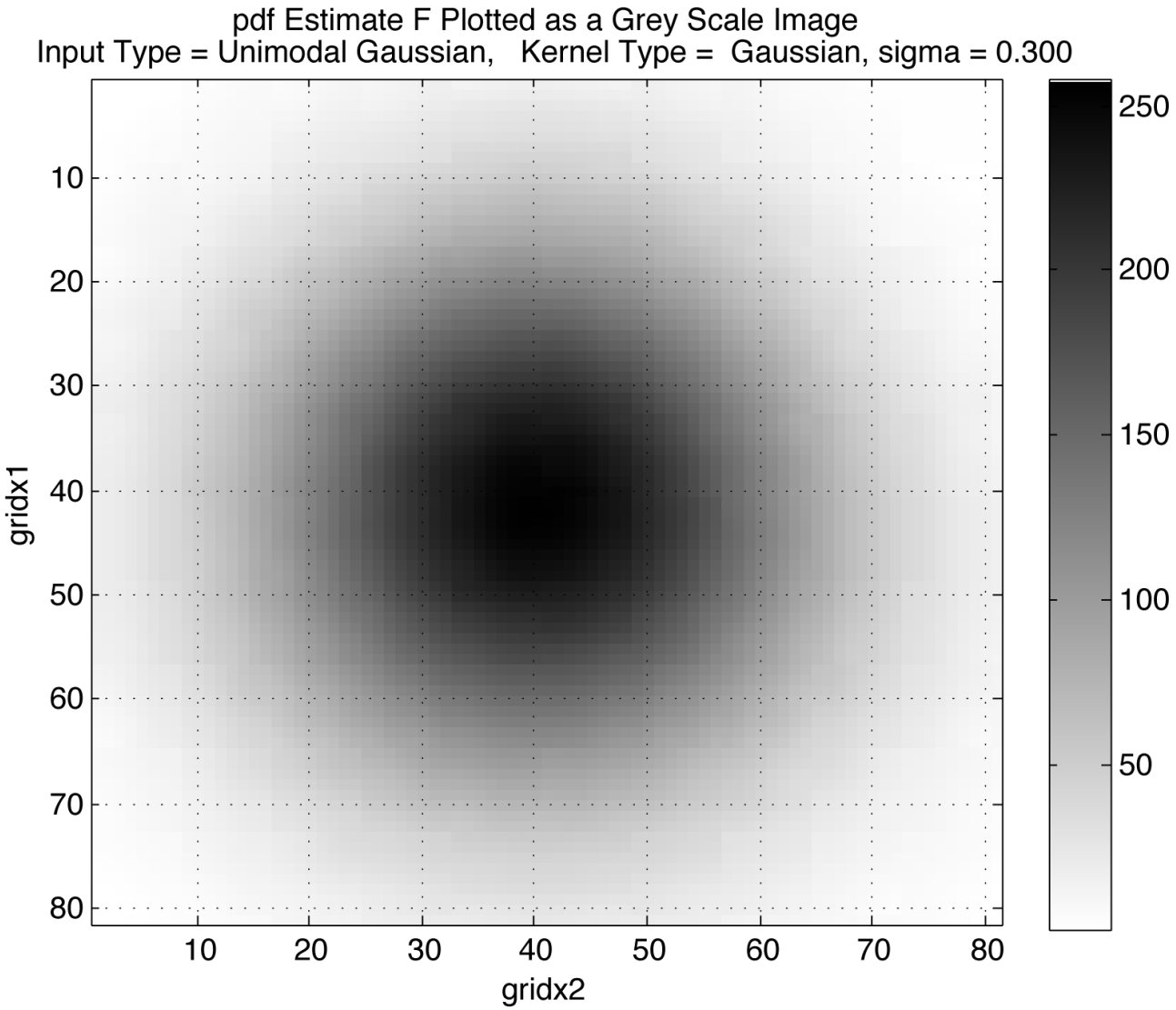


Figure 7.3: Example 1, X0: The kernel density pdf estimate $f(X0)$ for data set X0 is displayed as a grey scale image in this figure. The estimator used a Gaussian kernel with smoothing parameter $\sigma = 0.3$. The 2D grid for the pdf estimate is denoted $(gridx1, gridx2)$.

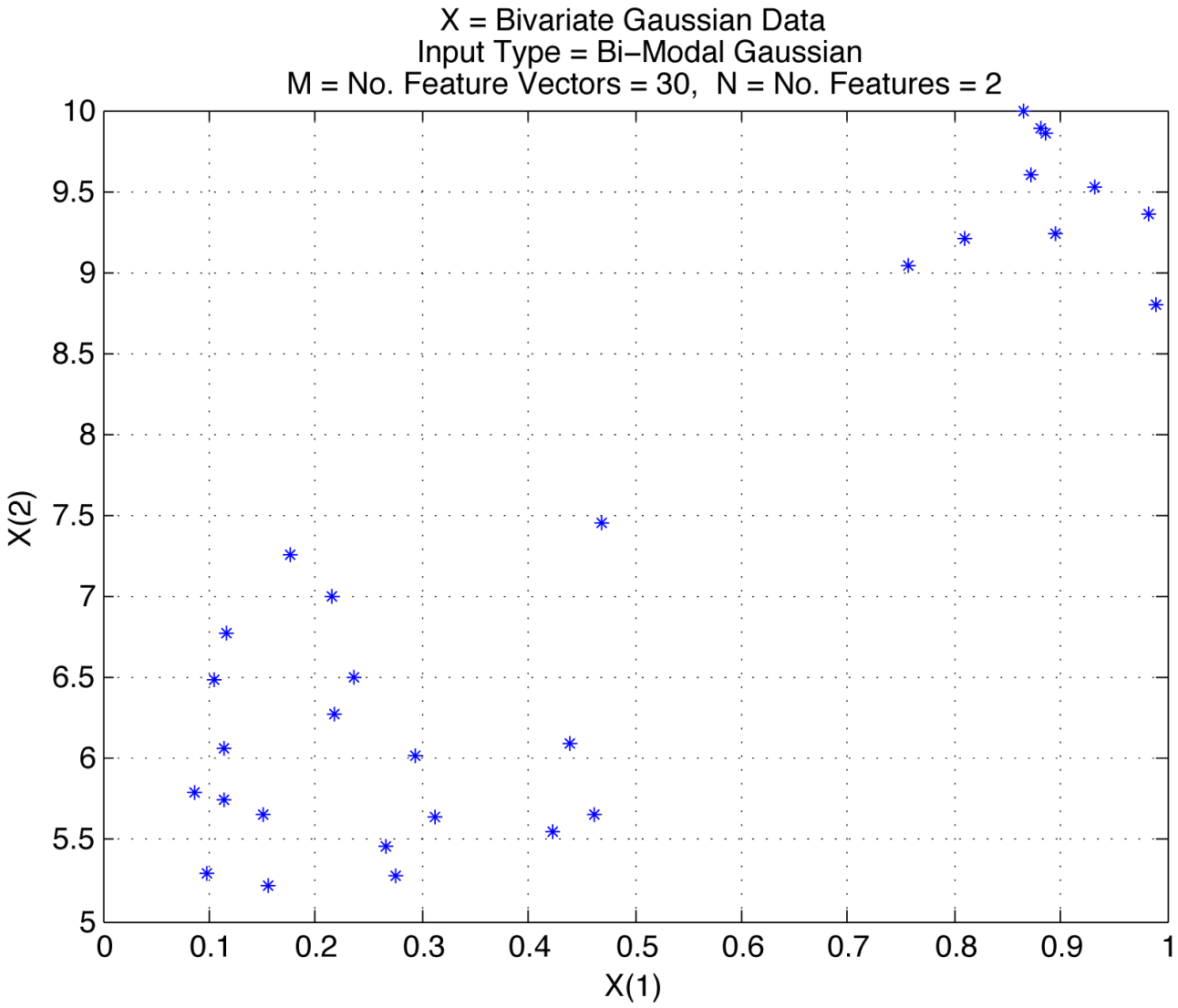


Figure 7.4: Example 1, X1: Scatter plot of the simulated observation data for bimodal bivariate Gaussian-distributed random variable X1. The data matrix for this plot is given in Equation (7.1).The abscissa $X(1)$ corresponds to the first column of the data matrix X, and the ordinate $X(2)$ corresponds to the second column of the data matrix X.

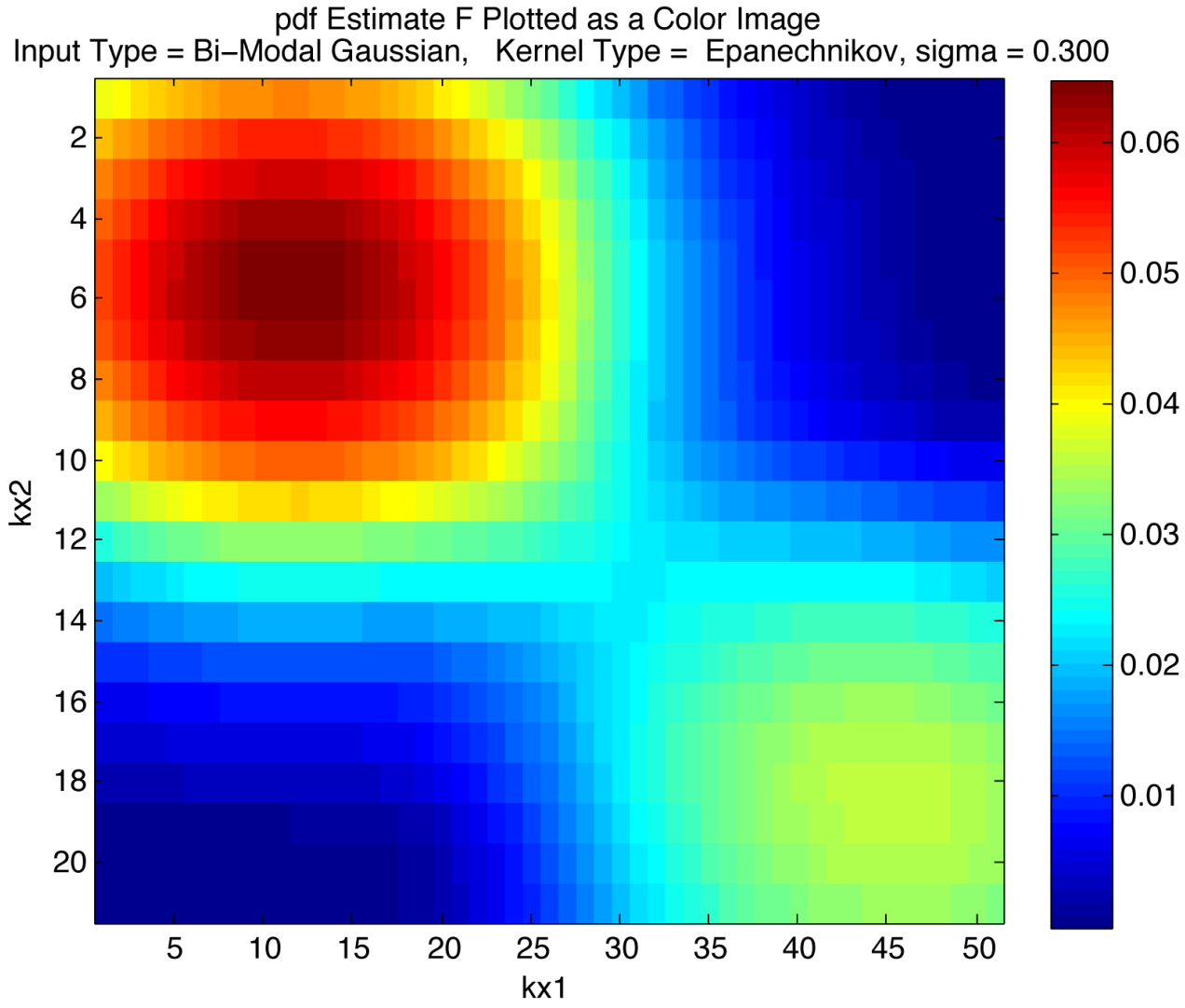


Figure 7.5: Example 1, X1: Color plot of the pdf estimate $f(X1)$ for the bimodal bivariate Gaussian random variable X1. The estimator used an Epanechnikov kernel with smoothing parameter $\sigma = 0.3$. The data matrix for this plot is given in Equation (7.1). The 2D grid for the pdf estimate is denoted $(kx1, kx2)$.

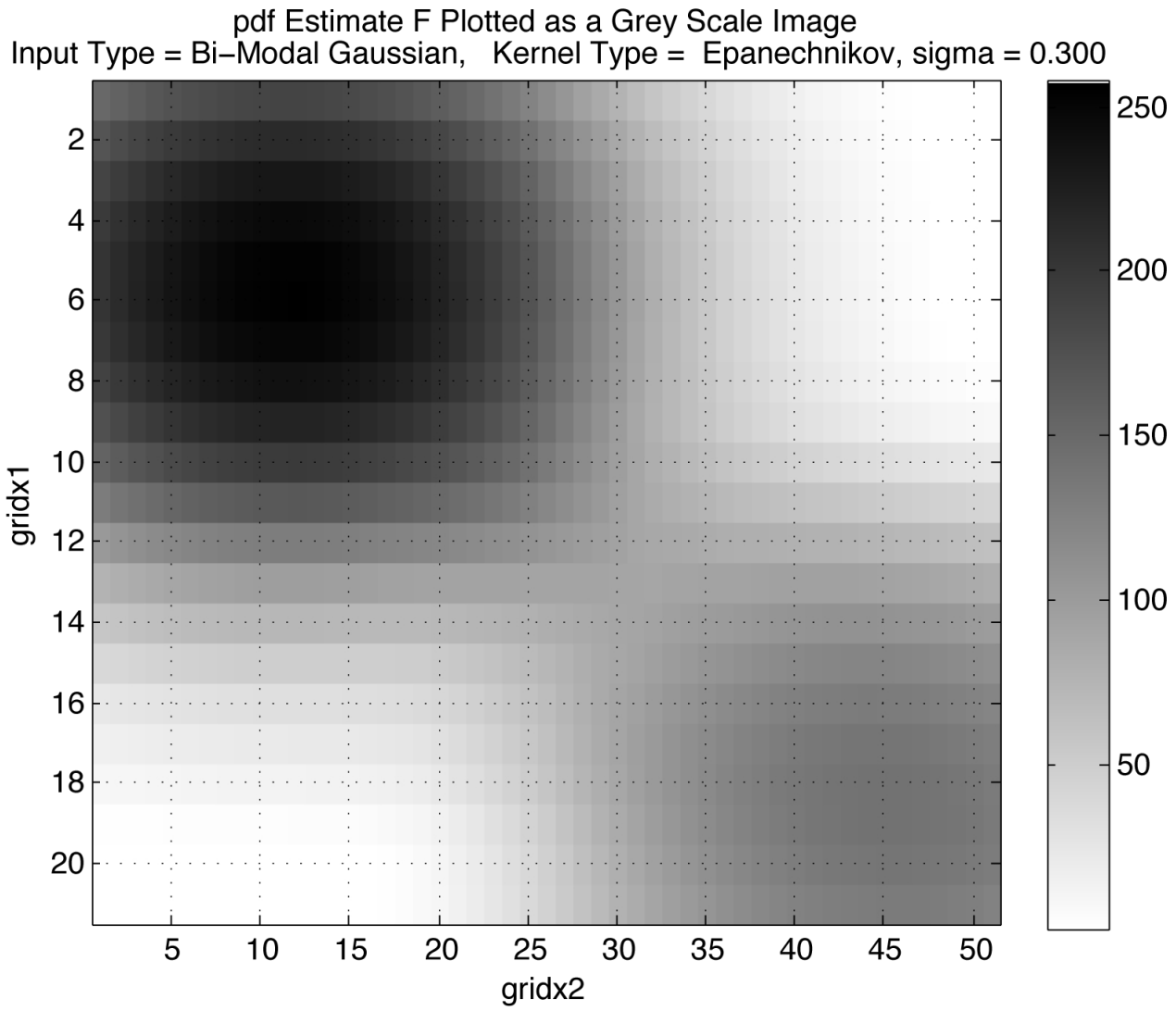


Figure 7.6: Example 1, X1: The kernel density pdf estimate $f(X1)$ for data set X1 is displayed as a grey scale image in this figure. The estimator used an Epanechnikov kernel with smoothing parameter $\sigma = 0.3$. The data matrix for this plot is given in Equation (7.1). The 2D grid for the pdf estimate is denoted $(gridx1, gridx2)$

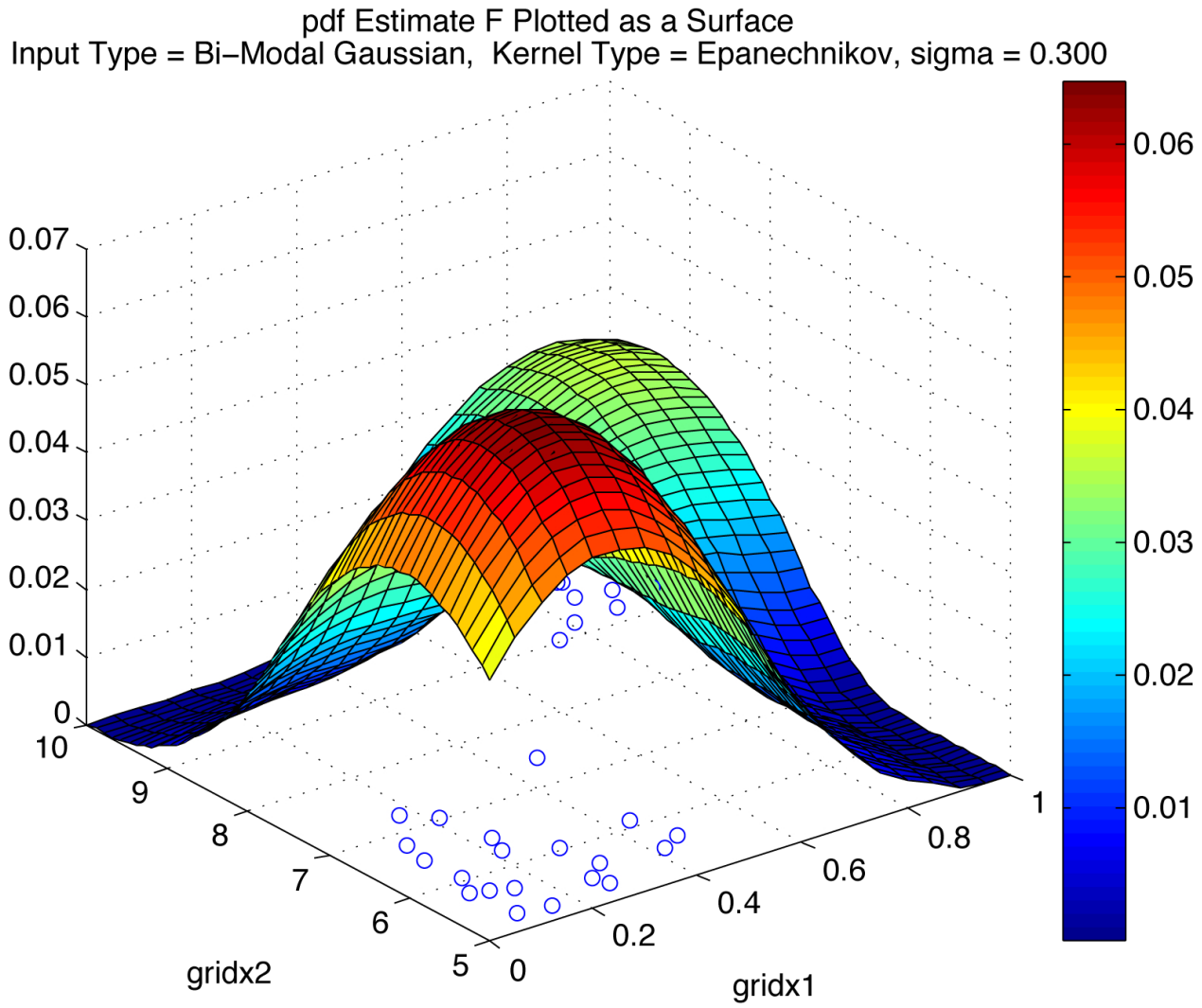


Figure 7.7: Example 1, X1: 3D plot of the pdf estimate $f(X1)$ of the bimodal bivariate Gaussian random variable X1. The estimator used an Epanechnikov kernel with smoothing parameter $\sigma = 0.3$. Note that the circles denote the original measurement samples from which the pdf estimate was calculated. The data matrix for this plot is given in Equation (7.1). The 2D grid for the pdf estimate is denoted $(gridx1, gridx2)$.

Chapter 8

Example 2: Simulation of 3D Random Variables, 3D pdf Estimation and 3D Hellinger Distance Computation

The purpose of this experiment is to demonstrate the performance of the new multi-dimensional pdf estimation code written for this work. In this experiment, we simulate two sets of data (labeled H0 and H1) for which we wish to estimate pdfs. Finally, after the pdf estimates are evaluated, we compute the Hellinger distance between the two pdfs of the two data sets.

8.1 Hypothesis H0: Simulated Trivariate Unimodal Gaussian Random Variable

This example uses simulated trivariate unimodal Gaussian training data vector $\underline{X}_{H_0} \sim N[\mu, \Sigma^{1/2}]$ with mean vector $\underline{\mu}$ and covariance matrix Σ defined as follows:

$$\underline{\mu} = [0 \;\; 0 \;\; 0]^T \qquad \text{Mean Vector} \tag{8.1}$$

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad \text{Covariance Matrix} \tag{8.2}$$

The kernel density estimator uses a Gaussian kernel with smoothing parameter $\sigma = .3$.

8.2 Hypothesis H1: Simulated Bimodal Non-Gaussian 3D Random Variable (Sum of Gaussians)

For Hypothesis H1, we simulate set of non-Gaussian 3-dimensional random vectors $\{\underline{X}_{H1}\}$ by summing two Gaussian r.v.'s: $\underline{X}_{H1} = \underline{X}_1 + \underline{X}_2$. The parameters for the pdf simulations are:

$$\underline{\mu_1} = [3 \;\; 3 \;\; 3]^T \qquad \text{Mean Vector 1} \tag{8.3}$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad \text{Covariance Matrix 1} \tag{8.4}$$

$$\underline{\mu_2} = [6 \;\; 6 \;\; 6]^T \qquad \text{Mean Vector 2} \tag{8.5}$$

$$\Sigma_2 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \qquad \text{Covariance Matrix 2} \tag{8.6}$$

The kernel density estimator uses an Epanechnikov kernel with smoothing parameter $\sigma = .3$.

8.3 Hellinger Distance Results

The calculated Hellinger distance between H0 and H1 for this example is $hD3D = 0.175$.

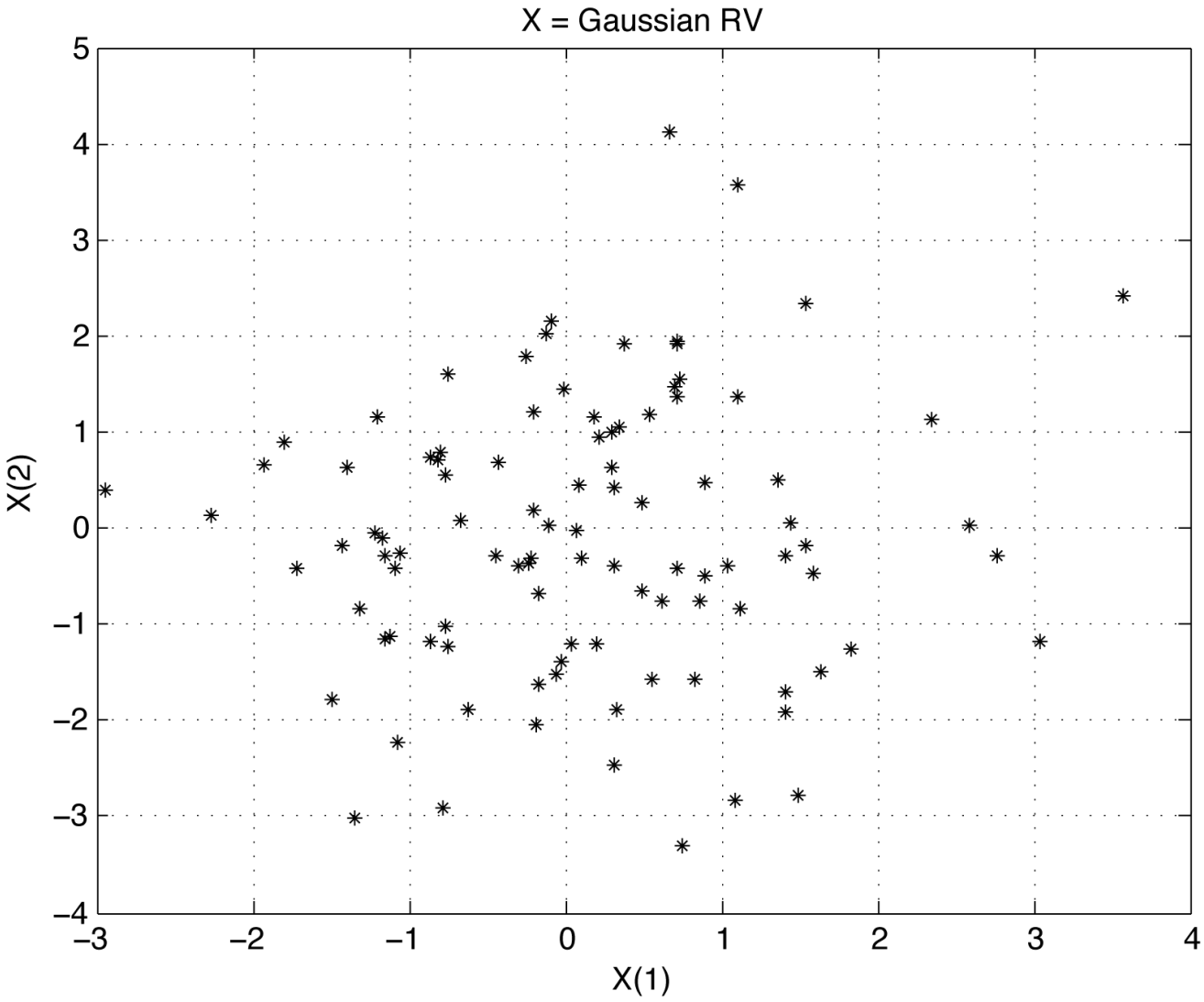


Figure 8.1: Fig 3D H0 X(1) X(2), Example 2, H0: Scatter plot of a slice of the simulated observation data $\underline{X}_{H_0}$ along dimensions 1 and 2 for the unimodal 3D Gaussian distributed random variable. The mean vector =$\underline{\mu}$, and the covariance matrix =Σ.

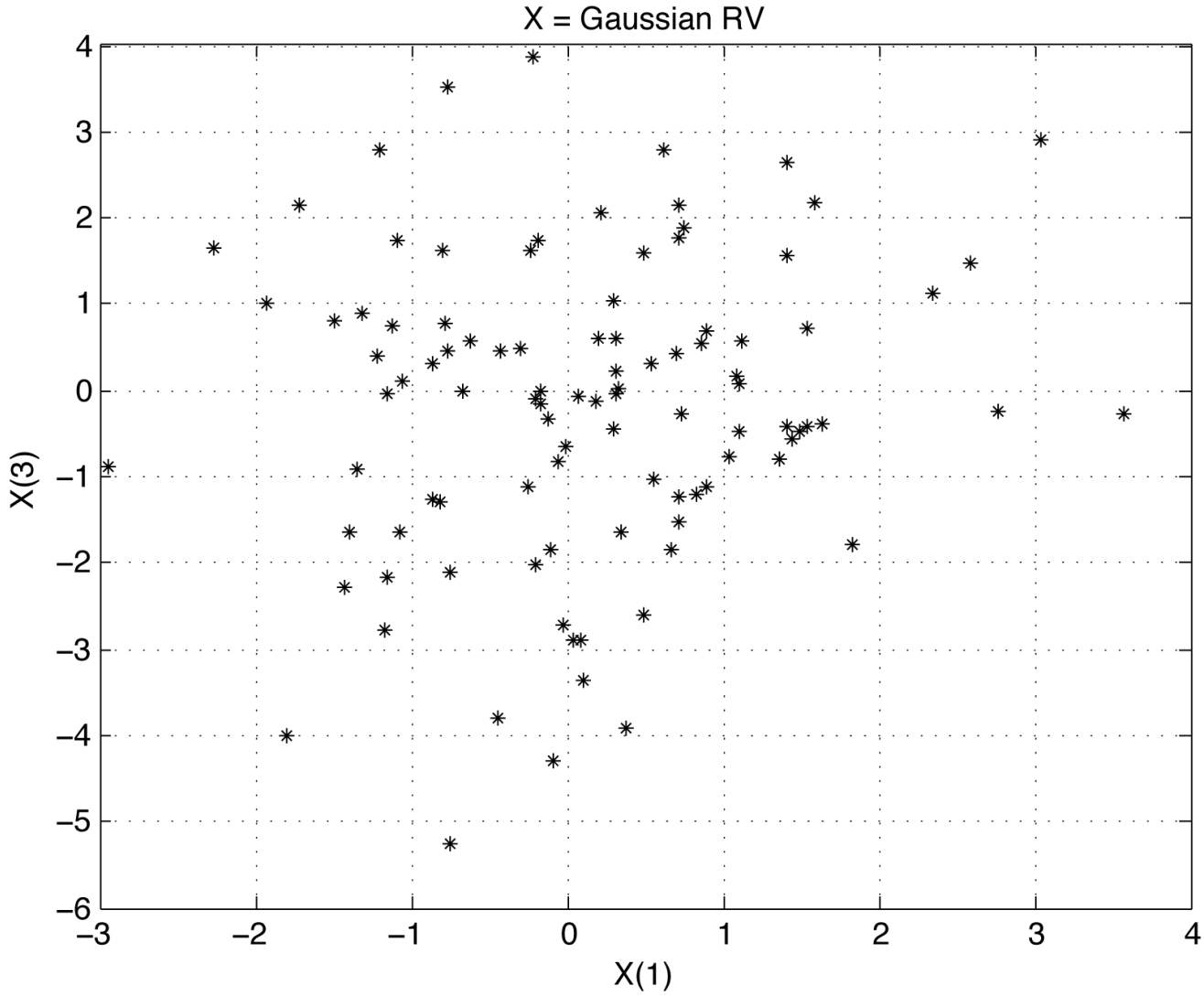


Figure 8.2: Fig 3D H0 X(1) X(3), Example 2, H0: Scatter plot of a slice of the simulated observation data $\underline{X}_{H_0}$ along dimensions 1 and 3 for the unimodal 3D Gaussian-distributed random variable H0. The mean vector =$\underline{\mu}$, and the covariance matrix $= \Sigma$.

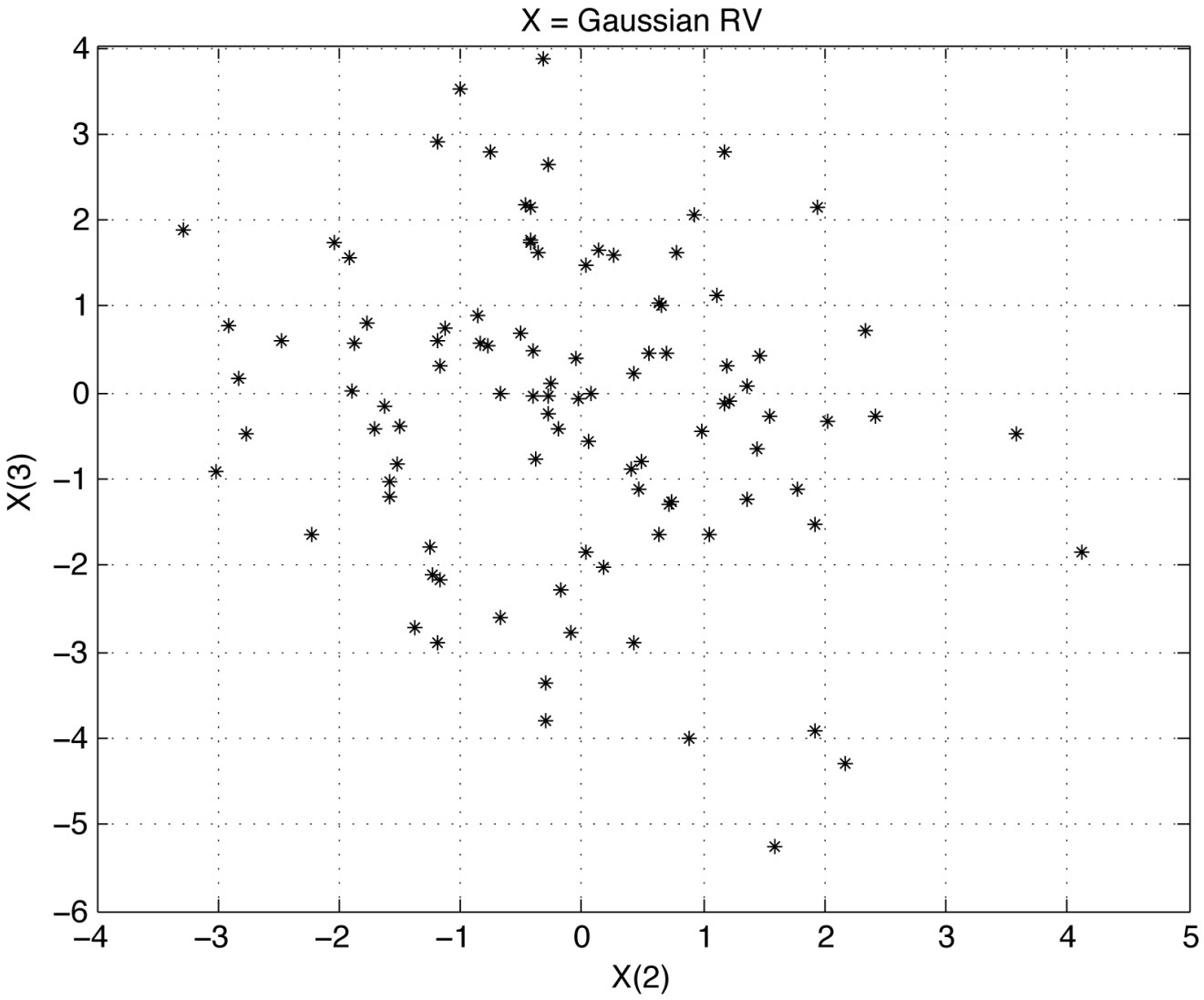


Figure 8.3: Fig 3D H0 X(2) X(3), Example 2, H0: Scatter plot of a slice of the simulated observation data $\underline{X}_{H_0}$ along dimensions 2 and 3 for the unimodal 3D Gaussian-distributed random variable H0. The mean vector =$\underline{\mu}$, and the covariance matrix = Σ.

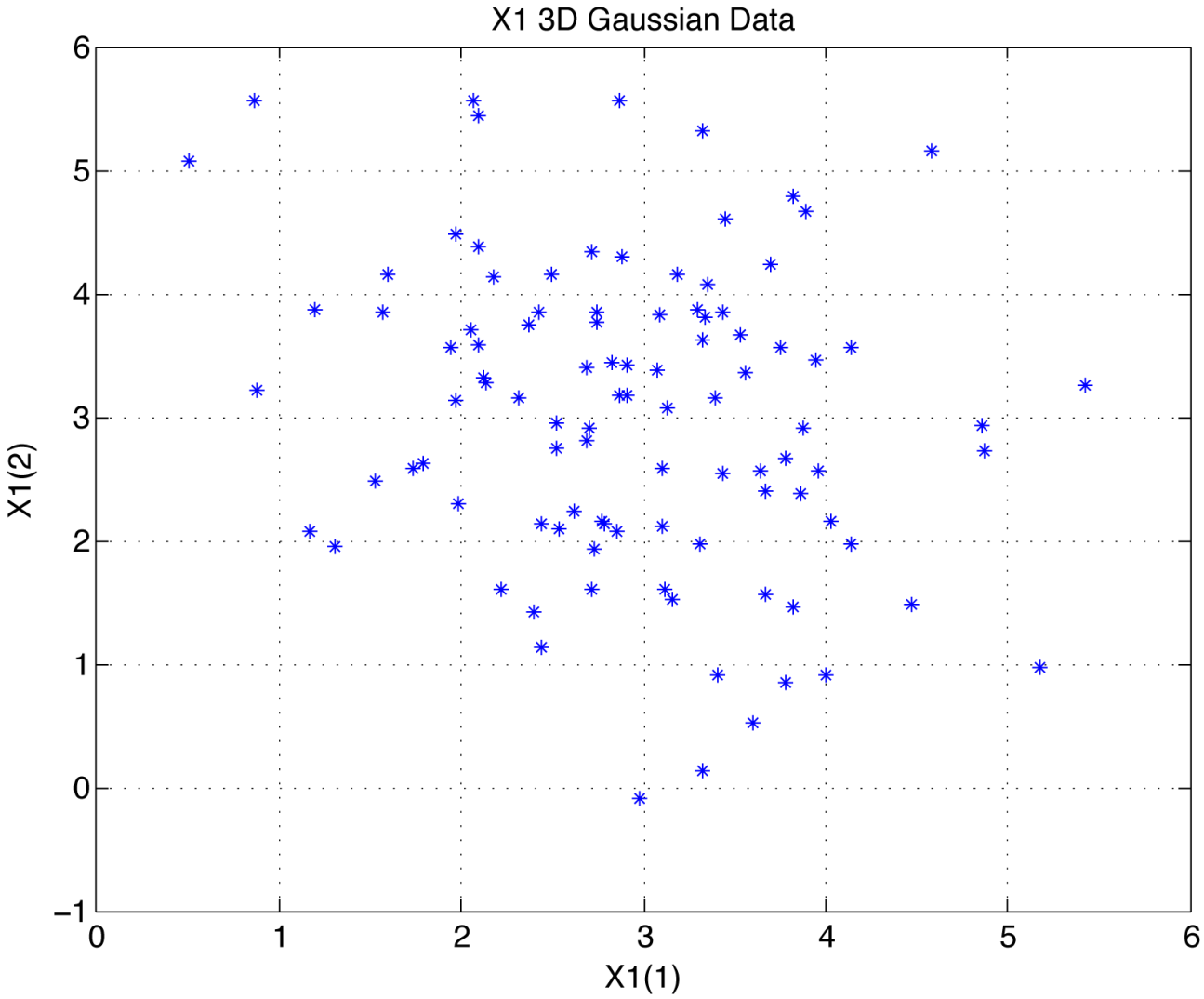


Figure 8.4: Fig 3D H1 X1(1) X1(2), Example 2, H1: Scatter plot of a slice of the simulated observation data $\underline{X}_{H_1}$ along dimensions X1(1) and X1(2) for the unimodal 3D Gaussian-distributed random variable H1. The mean vector =$\underline{\mu}_1$, and the covariance matrix $= \Sigma_1$.

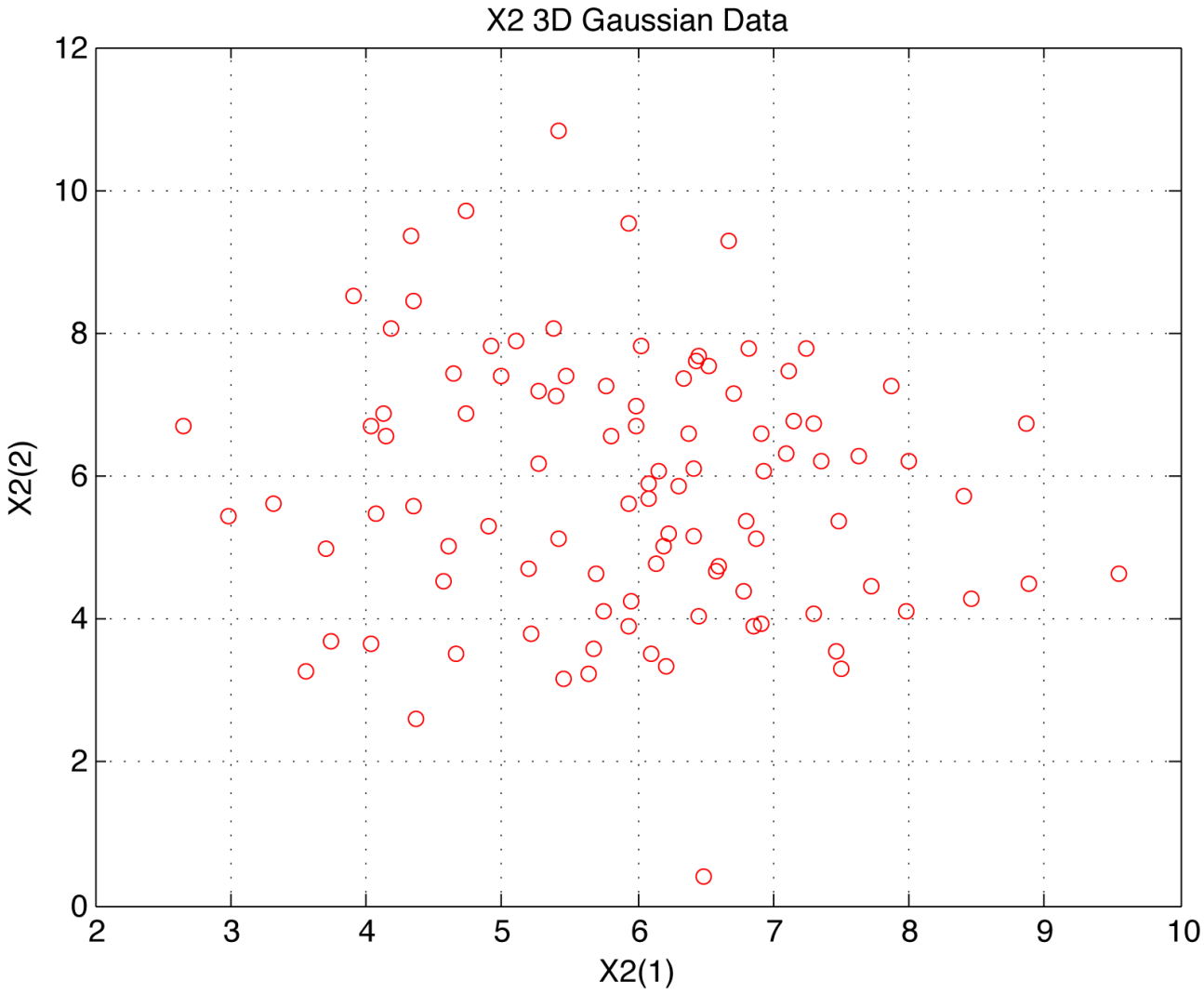


Figure 8.5: Fig 3D H1 X2(1) X2(2), Example 2, H1: Scatter plot of a slice of the simulated observation data $\underline{X}_{H_2}$ along dimensions X2(1) and X2(2) for the unimodal 3D Gaussian-distributed random variable H1. The mean vector =$\underline{\mu}_2$, and the covariance matrix $= \Sigma_2$.

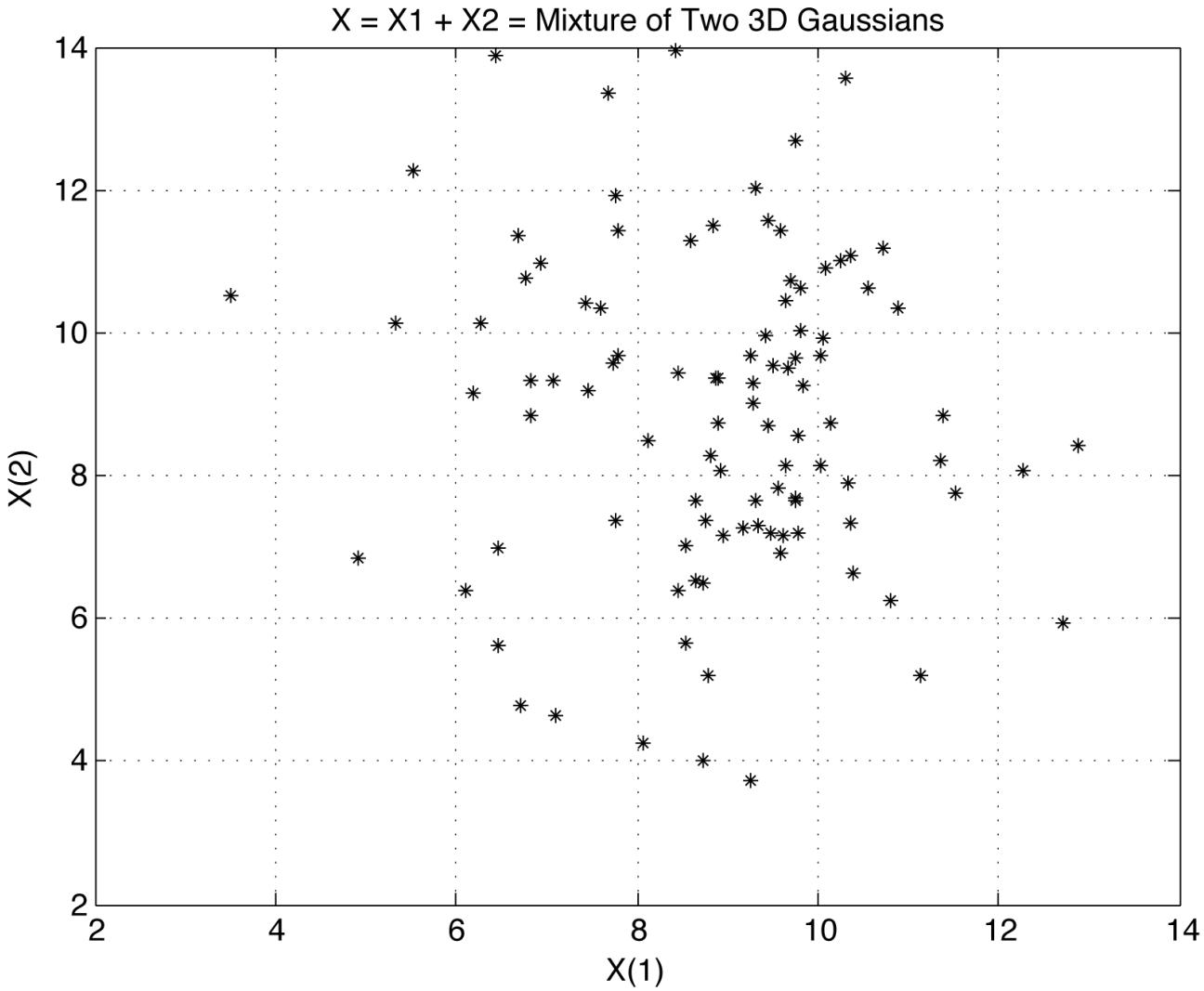


Figure 8.6: Fig 3D H1 X(1) X(2), Example 2, H1: Scatter plot of a slice of the set of simulated observation data vectors $\{\underline{X}_{H_1} = \underline{X}_1 + \underline{X}_1\}$ along dimensions X(1) and X(2) for the bimodal 3D random variable $\underline{X}_{H_1}$.

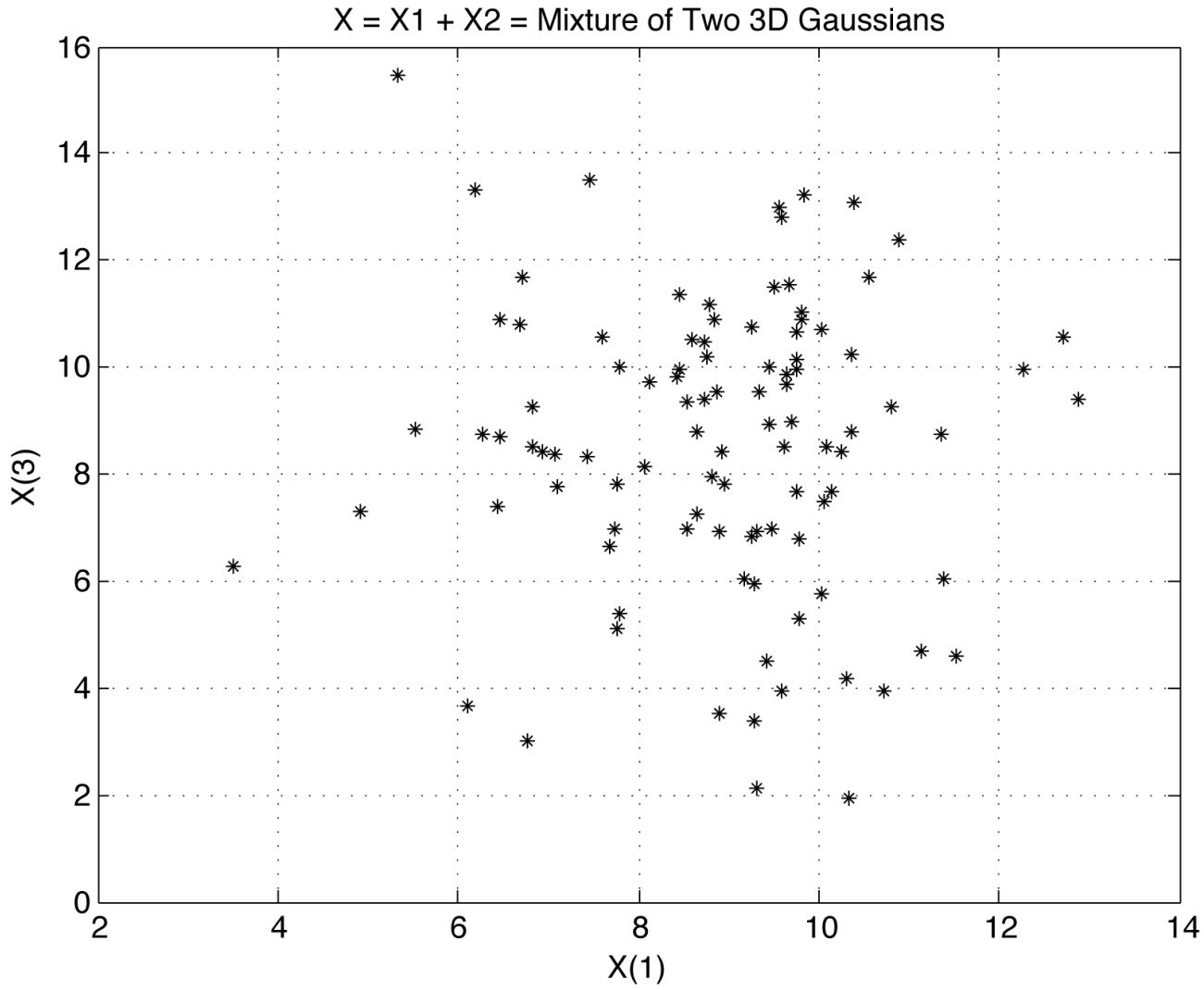


Figure 8.7: Fig 3D H1 X(1) X(3), Example 2, H1: Scatter plot of a slice of the set of simulated observation data vectors $\{\underline{X}_{H_1} = \underline{X}_1 + \underline{X}_1\}$ along dimensions X(1) and X(2) for the bimodal 3D random variable $\underline{X}_{H_1}$.

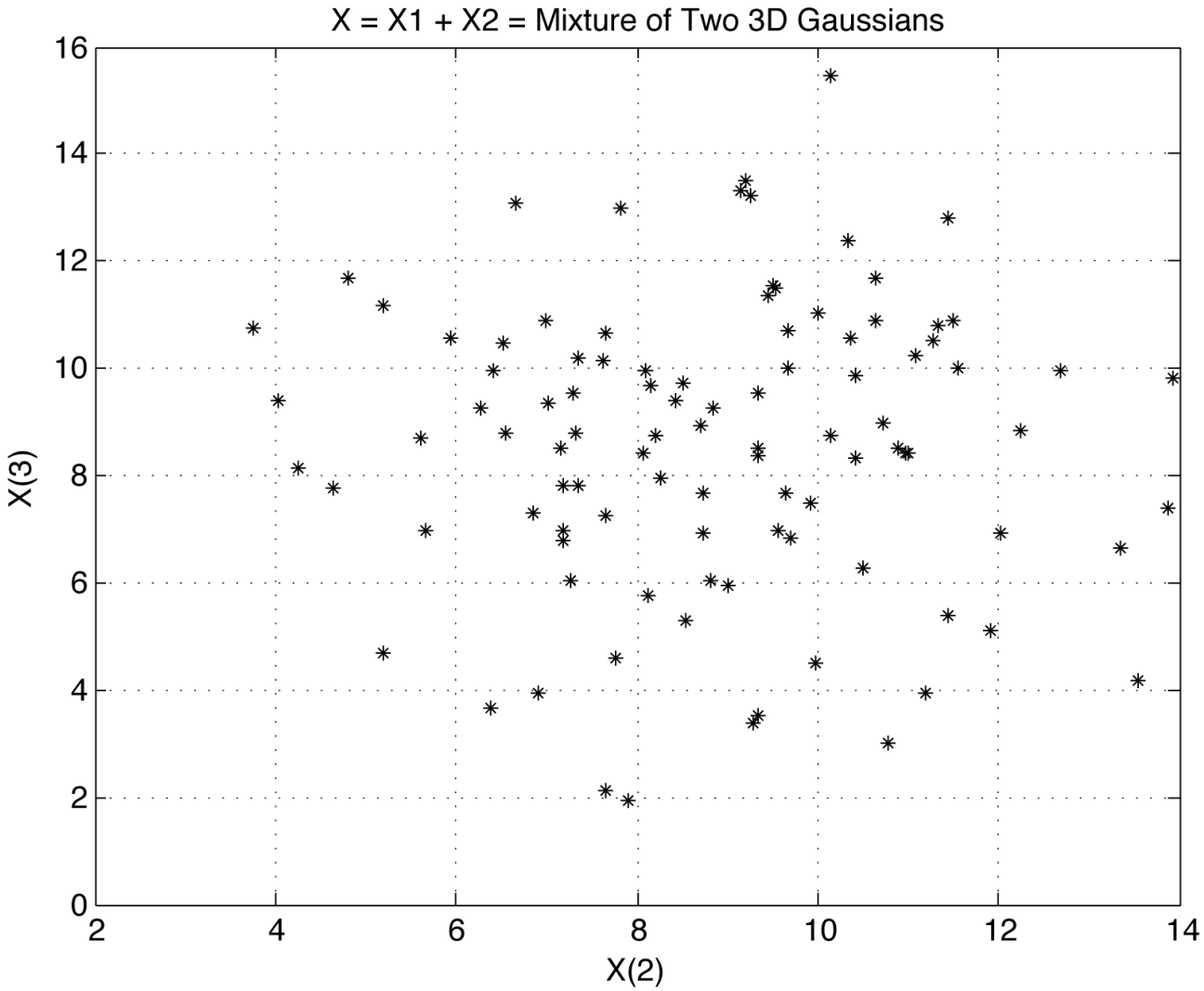


Figure 8.8: Fig 3D H1 X(2) X(3), Example 2, H1: Scatter plot of a slice of the set of simulated observation data vectors $\{\underline{X}_{H_1} = \underline{X}_1 + \underline{X}_1\}$ along dimensions X(1) and X(2) for the bimodal 3D random variable $\underline{X}_{H_1}$.

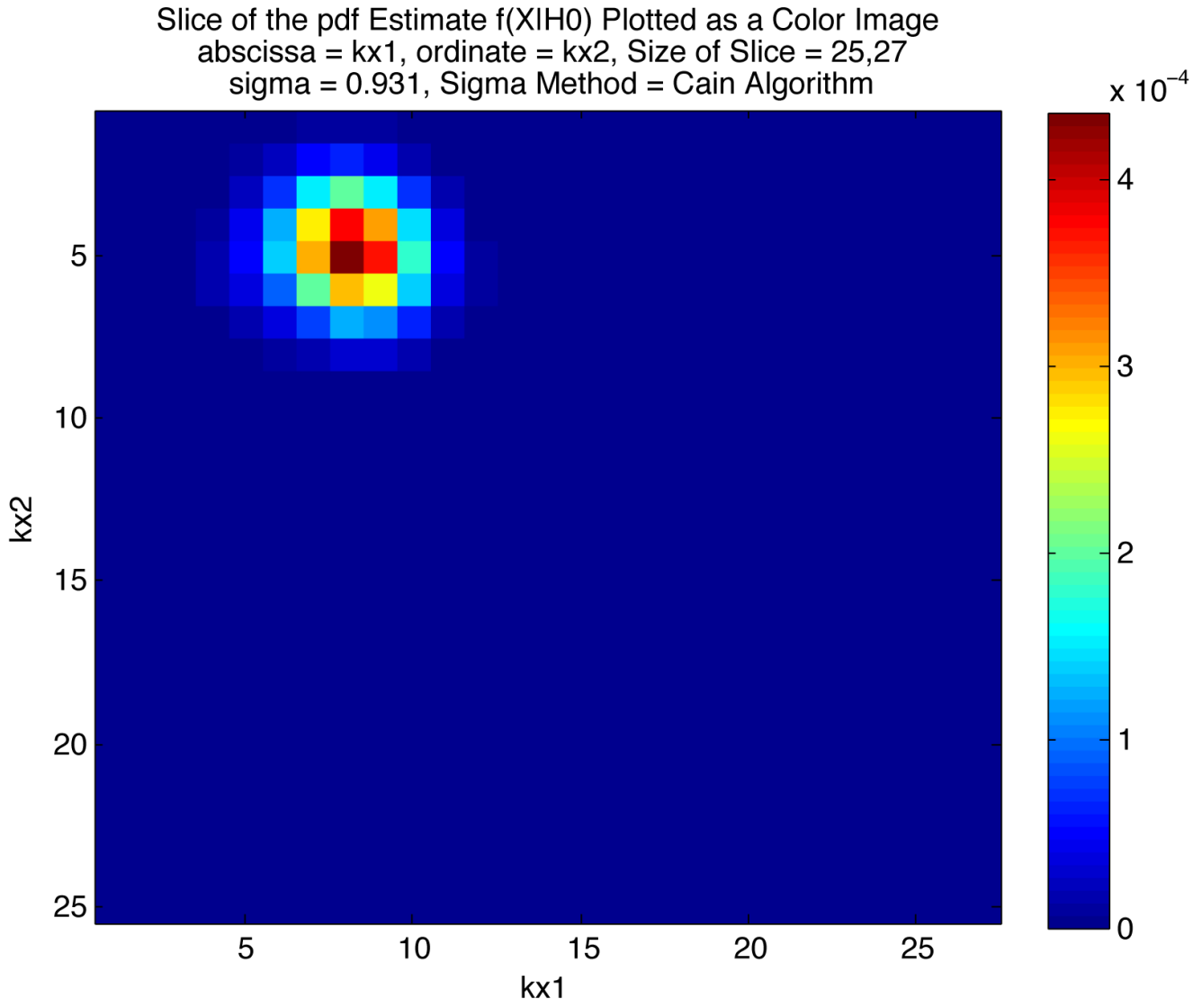


Figure 8.9: Fig 3D H0 X(kx1) X(kx2), Example 2, H0: Slice of the pdf estimate $f(\underline{X}_{H_0})$ of the random variable $\underline{X}_{H_0}$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

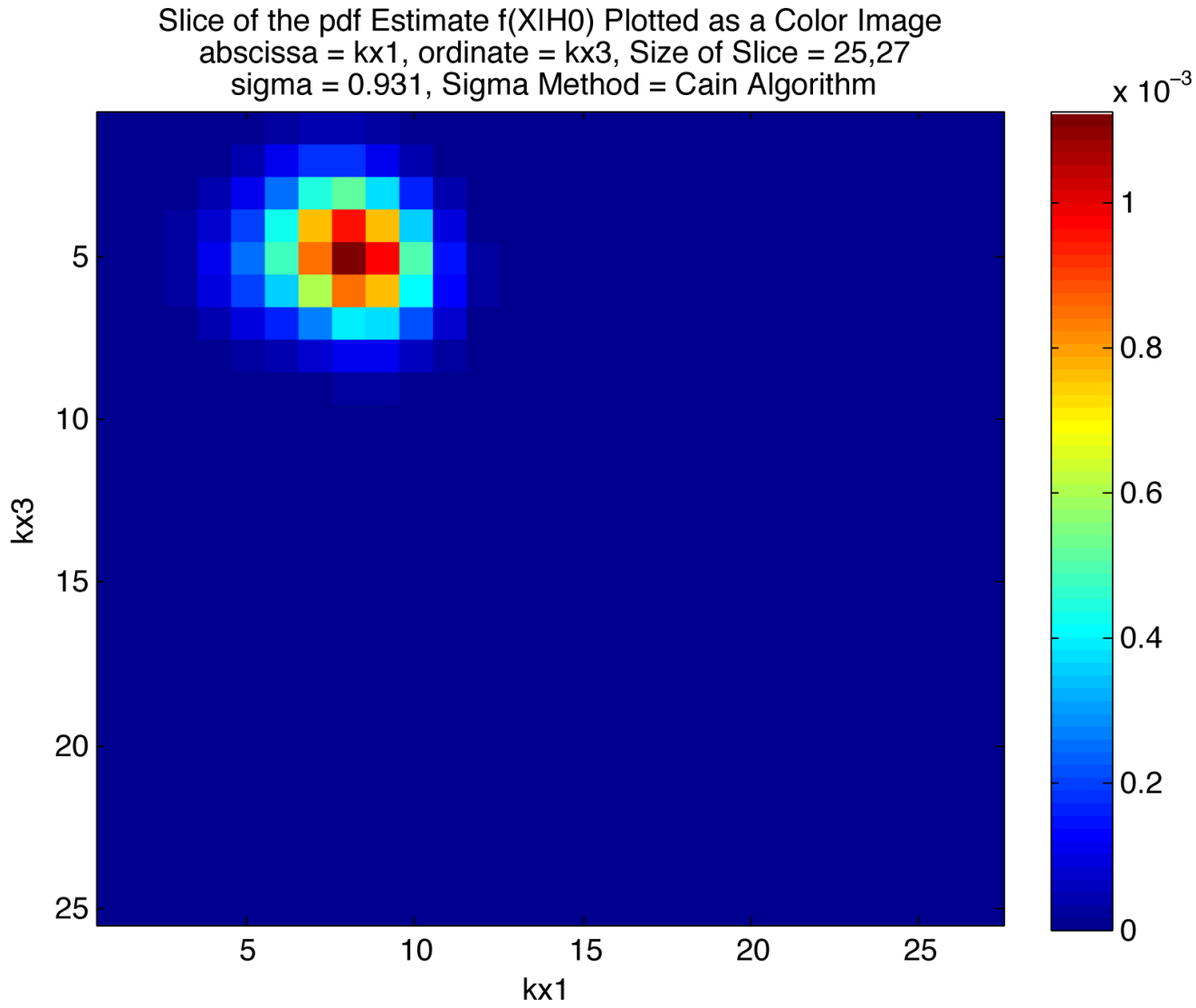


Figure 8.10: Fig 3D H0 X(kx1) X(kx3), Example 2, H0: Slice of the pdf estimate $f(\underline{X}_{H_0})$ of the random variable $\underline{X}_{H_0}$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

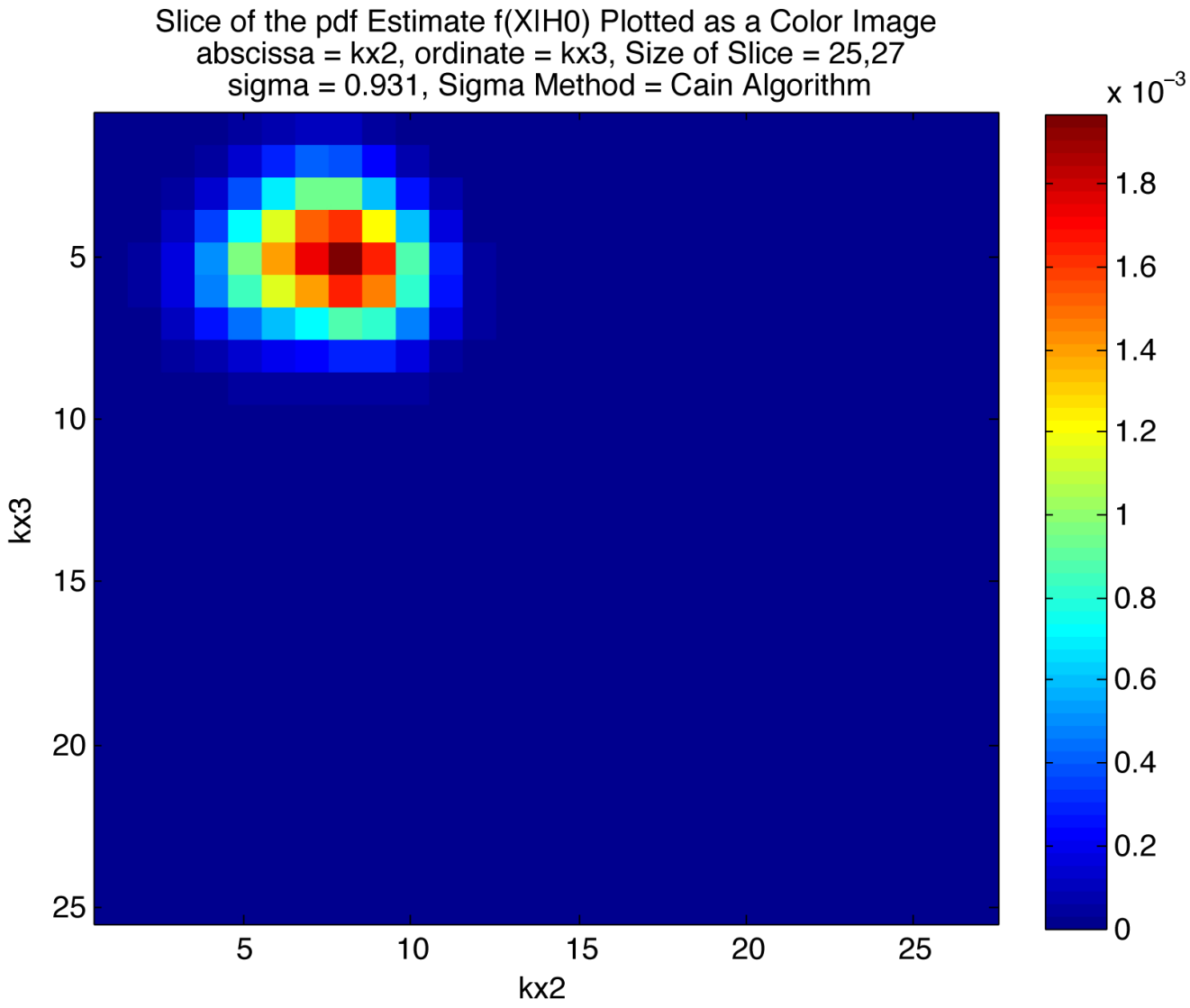


Figure 8.11: Fig 3D H0 X(kx2) X(kx3), Example 2, H0: Slice of the pdf estimate $f(\underline{X}_{H_0})$ of the random variable $\underline{X}_{H_0}$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

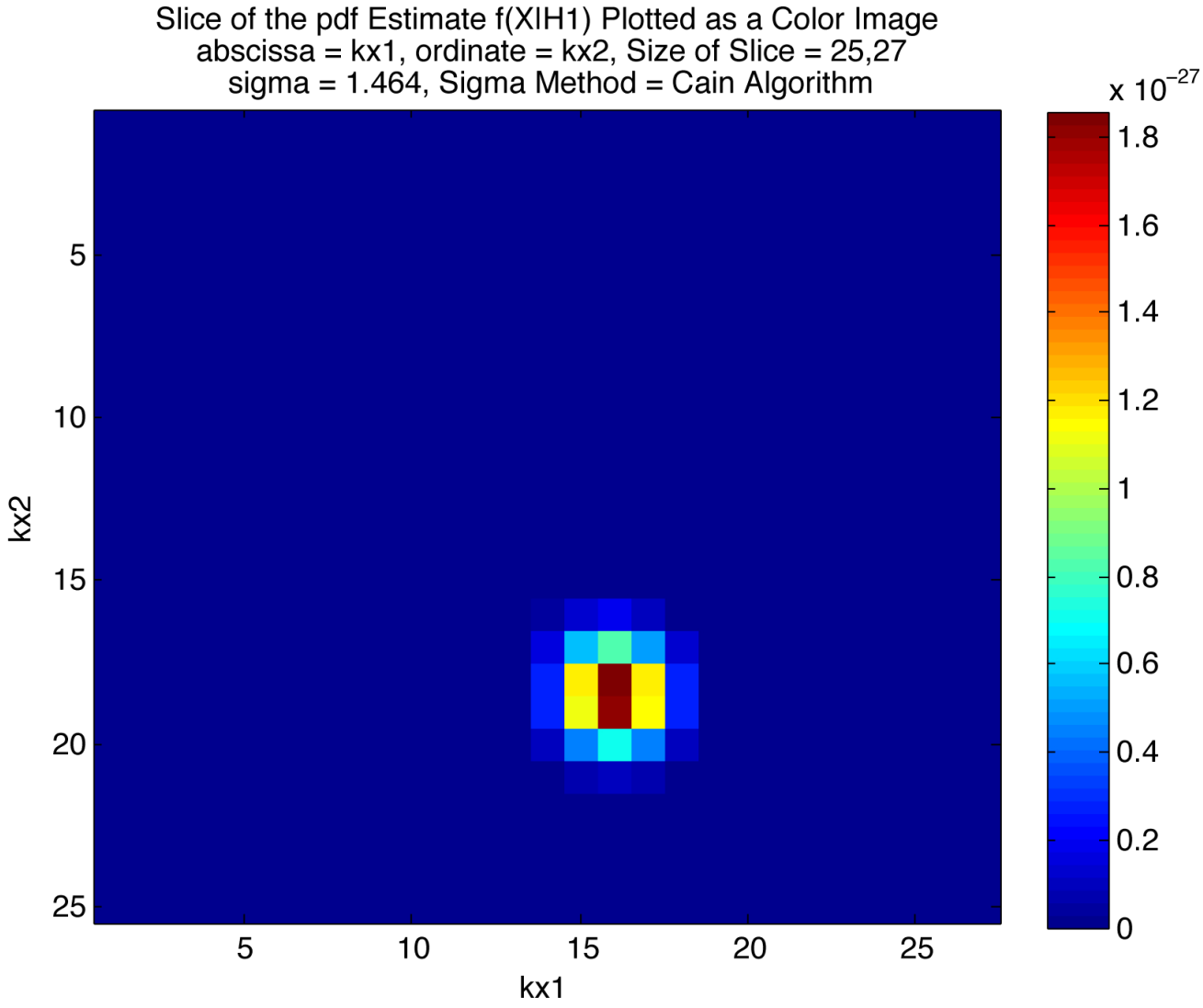


Figure 8.12: Fig 3D H1 X(kx1) X(kx2), Example 2, H1: Slice of the pdf estimate $f(\underline{X}_{H_1})$ of the random variable $\underline{X}_{H1} = \underline{X}_1 + \underline{X}_2$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

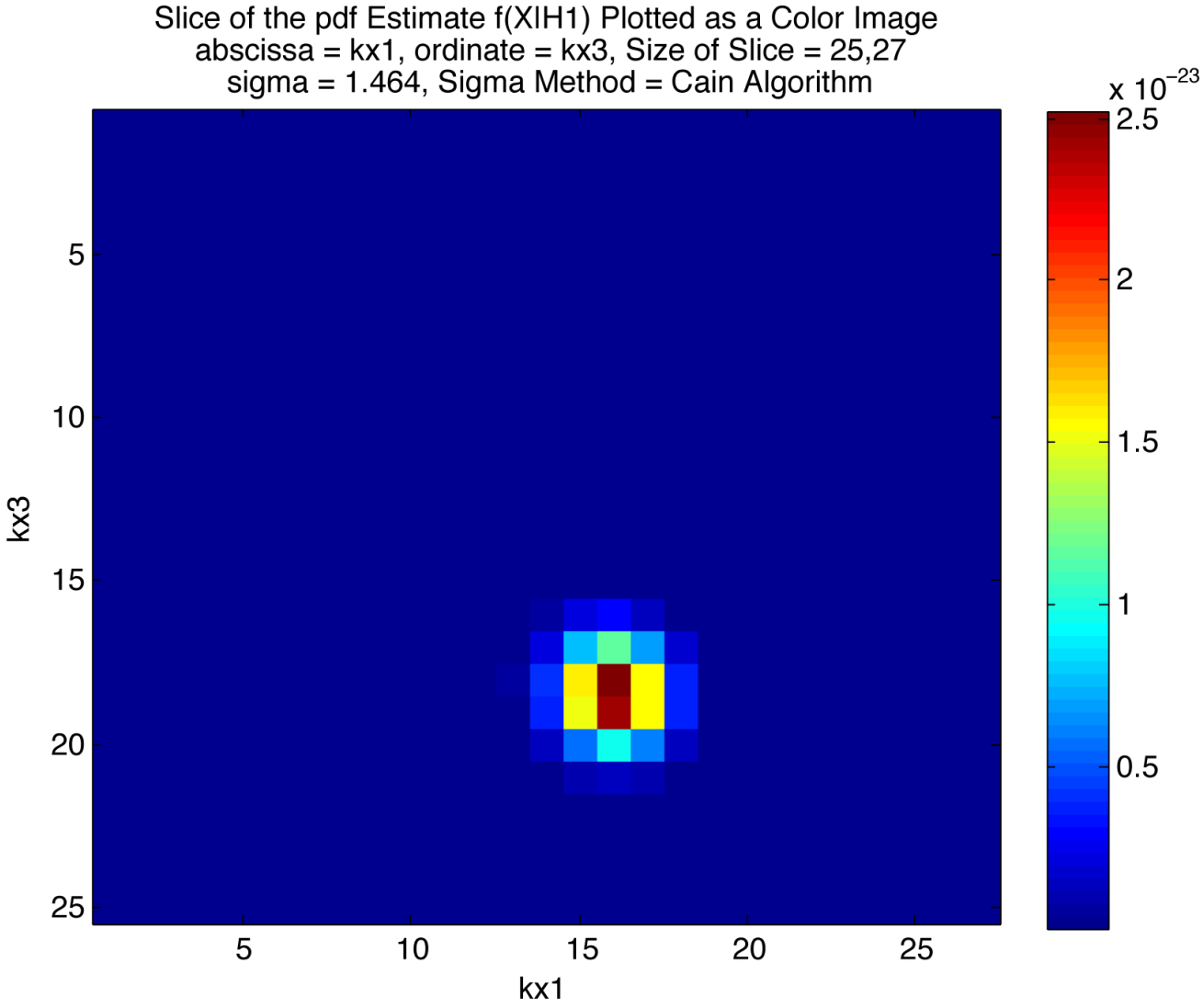


Figure 8.13: Fig 3D H1 X(kx1) X(kx3), Example 2, H1: Slice of the pdf estimate $f(\underline{X}_{H_1})$ of the random variable $\underline{X}_{H1} = \underline{X}_1 + \underline{X}_2$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

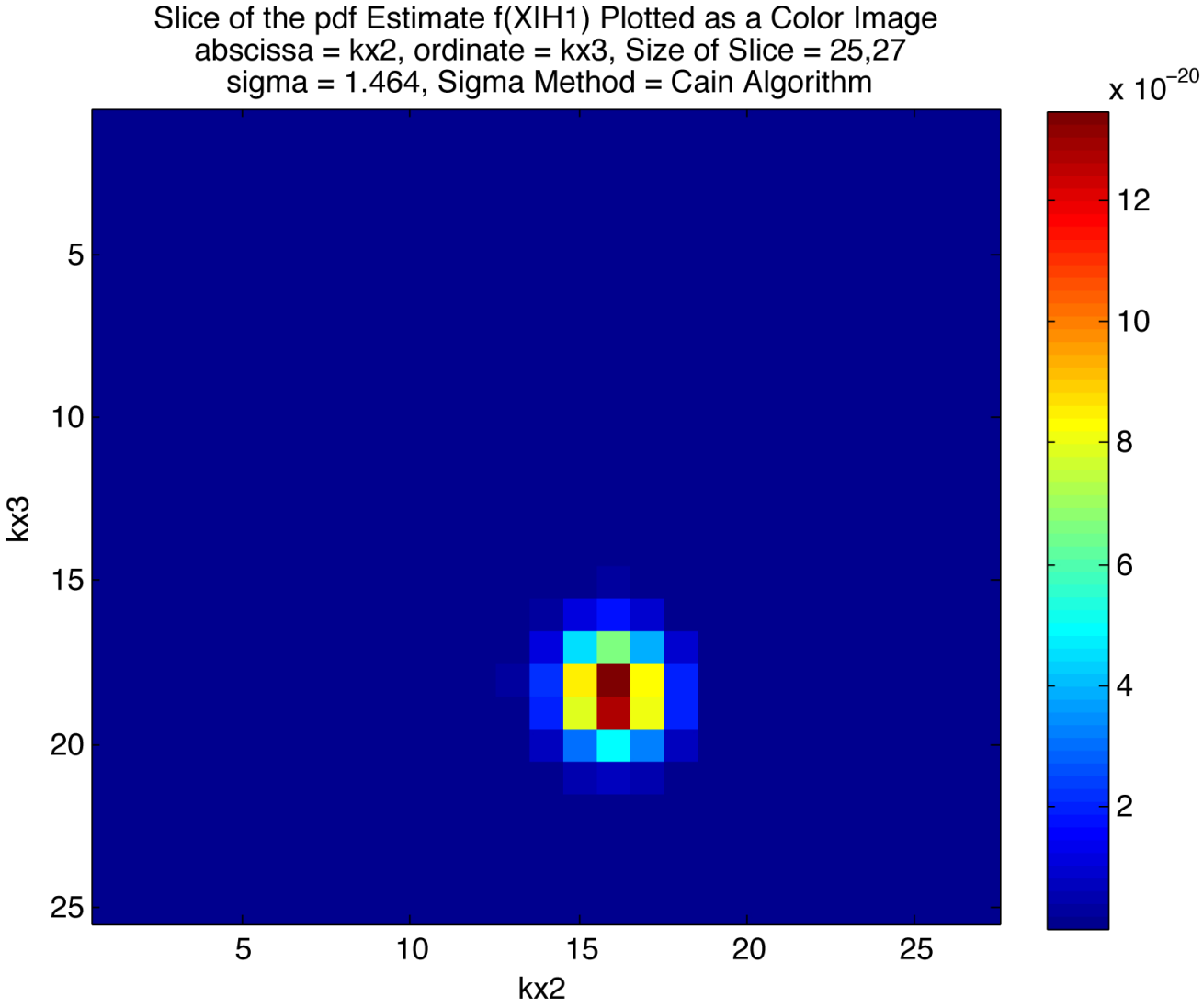


Figure 8.14: Fig 3D H1 X(kx2) X(kx3), Example 2, H1: Slice of the pdf estimate $f(\underline{X}_{H_1})$ of the random variable $\underline{X}_{H1} = \underline{X}_1 + \underline{X}_2$ plotted as a color image. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

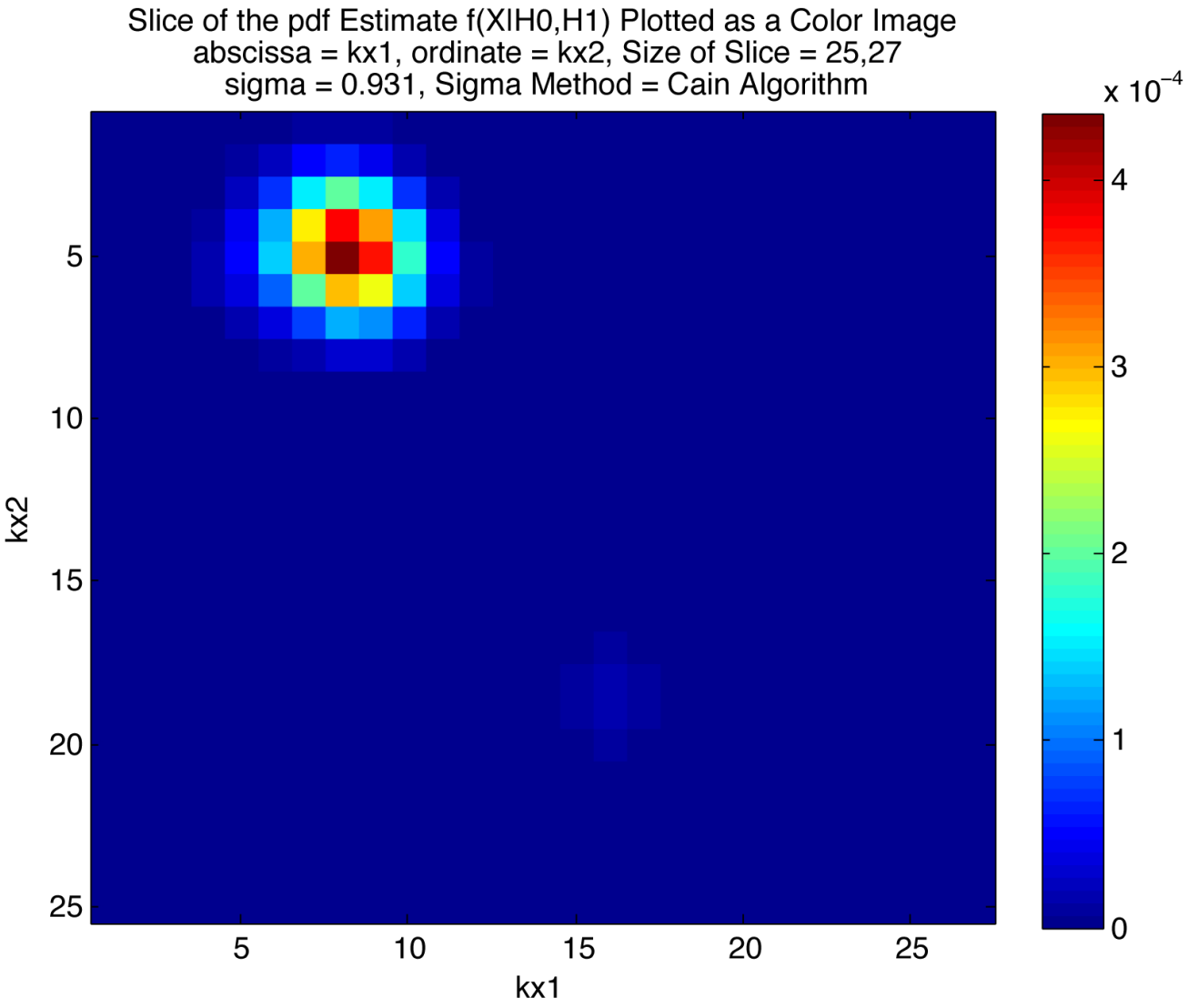


Figure 8.15: Fig 3D H0 H1 X(kx1) X(kx2), Example 2, H0 and H1: Slice of the joint pdf estimate $f(\underline{X}_{H_0}, \underline{X}_{H_1})$ plotted as a color image. This plot allows us to visualize the relationship between the two conditional pdf estimates $f(\underline{X}_{H_0})$ and $f(\underline{X}_{H_1})$. Unfortunately, in this example, the magnitude scales of the two pdfs are different enough that it is difficult to find one color scale that allows both pdfs to be visualized clearly. The pdf $f(\underline{X}_{H_1})$ is much fainter than $f(\underline{X}_{H_0})$ in this slice. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

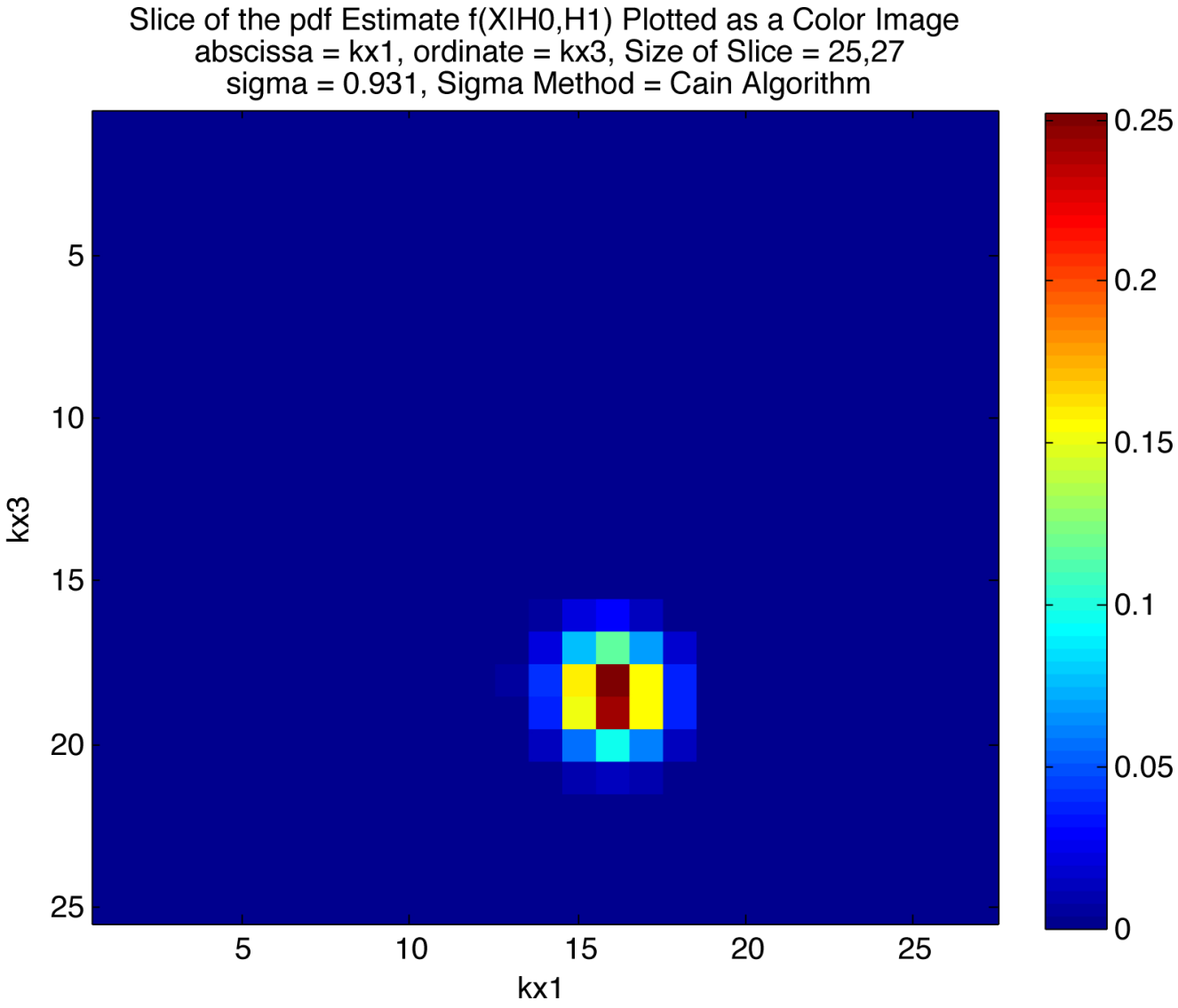


Figure 8.16: Fig 3D H0 H1 X(kx1) X(kx3), Example 2, H0 and H1: Slice of the joint pdf estimate $f(\underline{X}_{H_0}, \underline{X}_{H_1})$ plotted as a color image. This plot allows us to visualize the relationship between the two conditional pdf estimates $f(\underline{X}_{H_0})$ and $f(\underline{X}_{H_1})$. Unfortunately, in this example, the magnitude scales of the two pdfs are different enough that it is difficult to find one color scale that allows both pdfs to be visualized clearly. The pdf $f(\underline{X}_{H_0})$ is much fainter than $f(\underline{X}_{H_1})$ in this slice. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

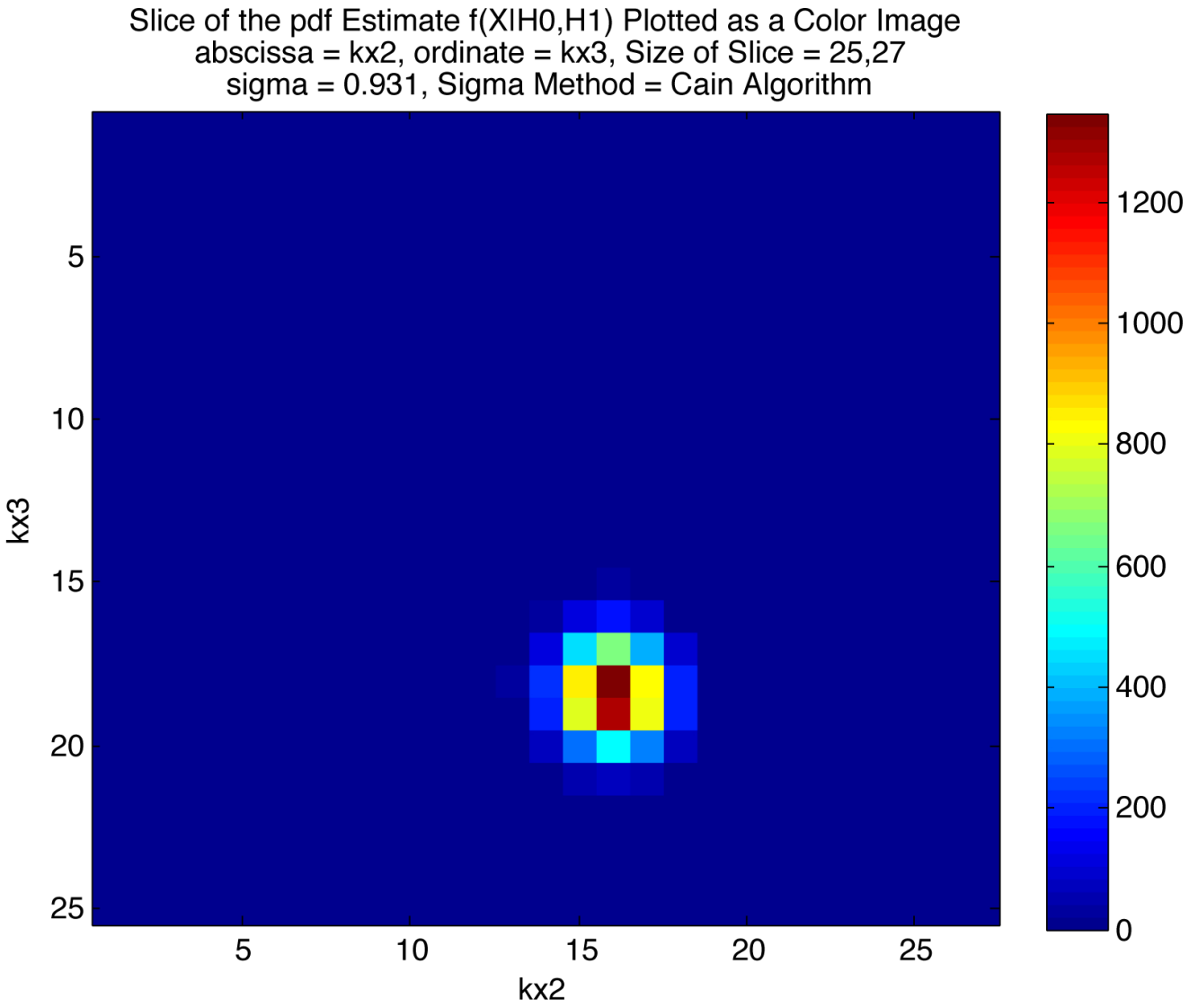


Figure 8.17: Fig 3D H0 H1 X(kx2) X(kx3), Example 2, H0 and H1: Slice of the joint pdf estimate $f(\underline{X}_{H_0}, \underline{X}_{H_1})$ plotted as a color image. This plot allows us to visualize the relationship between the two conditional pdf estimates $f(\underline{X}_{H_0})$ and $f(\underline{X}_{H_1})$. Unfortunately, in this example, the magnitude scales of the two pdfs are different enough that it is difficult to find one color scale that allows both pdfs to be visualized clearly. The pdf $f(\underline{X}_{H_0})$ is much fainter than $f(\underline{X}_{H_1})$ in this slice. The grid for the pdf estimate is denoted $(kx1, kx2, kx3)$.

Chapter 9

Conclusions

This report examines the use of information-theoretic class separability measures that can deal with non-Gaussian data sets. In particular, we show that the Hellinger distance (a divergence measure), has very desirable mathematical properties and propose that the Hellinger distance could be useful for feature selection when accompanied by a suitable density estimator and an optimal subset selection algorithm.

This report (1) Defines classification and feature selection theory, (2) Describes the continuous variable Hellinger distance in one dimension and discusses its mathematical properties in comparison with other distance measures, (3) Develops the discrete variable Hellinger distance expression for one and two dimensions, and shows that the discrete Hellinger distance for multiple dimensions is a straightforward extension, and (4) Presents experimental density estimation results using a new MATLAB implementation of a multivariate Parzen window kernel density estimator.

The draft of this report served as a starting point for Ensign Matthew Wilder's MSEE thesis work at the Naval Postgraduate School (NPS) in 2010 and 2011. Ensign Wilder completed his thesis and graduated from NPS in June 2011 (M. J. Wilder, "*Automatic Target Recognition: Statistical Feature Selection of Non-Gaussian Distributed Target Classes*," MSEE Thesis, Naval Postgraduate School, Thesis Advisor: Grace A. Clark, Second Reader: Monique P. Fargues, June 2011). In the thesis, Ensign Wilder created implementations of the Branch and Bound and Sequential Feature Selection algorithms, extended the multivariate kernel density estimator code created for this work, and demonstrated the performance of Hellinger Distance-based feature selection algorithms using both simulated data and data from a real-world benchmark data set. The new algorithm was shown to be very effective for both Gaussian and non-Gaussian data sets. The advantages gained for non-Gaussian data come at the cost of increased computational complexity. Using a laptop computer for practical problems, we generally use initial feature vectors containing no more than about ten or twelve features. Proposed future research lies in creating faster kernel density estimator algorithms that would allow for the use of higher dimensional data.

Bibliography

9.1 Classification and Feature Selection, Clark et al.

[1] M. J. Wilder and G. A. Clark, *Statistical Feature Selection for Non-Gaussian Distributed Target Classes*, 16th Annual Center for Advanced Signal and Imaging Sciences (CASIS) Signal and Imaging Sciences Workshop, Lawrence Livermore National Laboratory, May 23, 2012.

[2] M. J. Wilder, *Automatic Target Recognition: Statistical Feature Selection of Non-Gaussian Distributed Target Classes*, MSEE Thesis, Naval Postgraduate School, Thesis Advisor: Grace A. Clark, Second Reader: Monique P. Fargues, June 2011.

[3] G. A. Clark, C. L. Robbins, K. A. Wade and P. R. Souza, *Cable Damage Detection System and Algorithms Using Time Domain Reflectometry*, Lawrence Livermore National Laboratory report LLNL-TR-413970, June 16, 2009.

[4] G. A. Clark, S. K. Sengupta, W. D. Aimonetti, R. Roeske, J. G. Donetti, *Multispectral Image Feature Selection for Land Mine Detection*, IEEE Trans. Geoscience Remote Sensing, Vol. 38, No. 1, January 2000.

[5] G. A. Clark, M. C. Axelrod and D. D. Scott, *Classification of Heart Valve Sounds from Experiments in an Anechoic Water Tank*, Lawrence Livermore National Laboratory report, UCRL-JC-134634, June 1, 1999.

[6] F. Roeske, W. Aimonetti, G. Clark, J. Donetti, D. Fields, P. Schaich, S. K. Sengupta, R. J. Sherwood, *Preliminary Results of LLNL Analysis of NCSS Cobra Flight Test Data for Mine Detection*, Lawrence Livermore National Laboratory report, September, 1994.

[7] G. A. Clark, S. K. Sengupta, P. C. Schaich, R. J. Sherwood, M. R. Buhl, J. E. Hernandez, D. J. Fields, and M. R. Carter, *Data Fusion for the Detection of Buried Land Mines*, International Symposium on Substance Identification Technologies, European Optical Society (EOS)/The International Society for Optical Engineering (SPIE), Congress Centre Innsbruck, Innsbruck, Austria, October 4-8, 1993.

[8] G. A. Clark, Sailes K. Sengupta, Robert J. Sherwood, Jose E. Hernandez, Michael R. Buhl, Paul C. Schaich, Ronald J. Kane, Marvin J. Barth and Nancy K. DelGrande, *Sensor Feature Fusion for Detecting Buried Objects*, SPIEs 1993 International Symposium and Exhibition on Optical Engineering and Photonics in Aerospace and Remote Sensing, Conference on Underground and Obscured Imaging and Detection, Orlando, FL, April 12-16, 1993.

[9] C. A. Anderson, G. A. Clark, et. al., *LLNL Electro-Optical Mine Detection Program*, Livermore National Laboratory report UCRL-ID-118672, September, 1994.

[10] G. A. Clark, S. K. Sengupta, M. R. Buhl, R. J. Sherwood, P. C. Schaich, N. Bull, R. J. Kane, and M. J. Barth, *Detecting Buried Objects by Fusing Dual-Band Infrared Images*, Invited paper, Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, Nov. 1-3, 1993.

[11] M. E. Glinsky, G. A. Clark, Peter K.-Z. Cheng, K. R. Sandhya Devi, J., H. Robinson, and G. E. Ford, *Automatic Event Picking in Prestack Migrated Gathers Using a Probabilistic Neural Network*, Geophysics, Vol. 66, No. 5 (September-October, 2001), pp. 1488-1496.

[12] G. A. Clark, S. K. Sengupta, W. D. Aimonetti, F. Roeske and J. G. Donetti, *Multispectral Image Feature Selection for Land Mine Detection*, Lawrence Livermore National Laboratory report UCRL-JC-124375-Rev.1, IEEE Trans. Geoscience and Remote Sensing, January, 2000, pp. 304-311.

[13] G. A. Clark, *The Revelations of Acoustic Waves*, Science and Technology Review, Lawrence Livermore National Laboratory, Lawrence Livermore National Laboratory, UCRL-52000-99-5, U. S. Government Printing Office 1999/783-046-80013, May, 1999.

[14] P. C. Schaich, G. A. Clark, K.-P. Ziock, S. K. Sengupta, *Automatic Image Analysis for Detecting and Quantifying Gamma-Ray Sources in Coded Aperture Images*, IEEE Transactions on Nuclear Science, Vol. 43, No. 4, August, 1996.

[15] G. A. Clark, D. D. Scott, et. al., *Heart Valve Classification Using Opening and Closing Sounds*, Imaging Sciences Workshop, Center for Advanced Signal and Imaging Sciences, Lawrence Livermore National Laboratory, November 21-22, 1996.

[16] G. A. Clark, M. E. Glinsky, K. R. S. Devi, J. H. Robinson, P. K.-Z. Cheng, G. E. Ford, *Automatic event picking in pre-stack migrated gathers using a probabilistic neural network*, Society of Exploration Geophysicists (SEG) International Exposition and 66th Annual Meeting, Denver, Colorado, November 10-15, 1996.

[17] G. A. Clark, B. C. Bowman, N. Boruta and G. H. Thomas, *Classification of Heart Valve Condition Using Acoustic Measurements*, Imaging Sciences Workshop, Center for Signal and Imaging Sciences, Lawrence Livermore National Laboratory, November 15-16, 1994.

[18] G. A. Clark, M. C. Axelrod and D. D. Scott, *Classification of Heart Valve Sounds from Experiments in an Anechoic Water Tank* Lawrence Livermore National Laboratory report UCRL-JC-134634, June 1, 1999.

[19] B. C. Bowman, G. A. Clark, G. H. Thomas, and N. Boruta, *Classification of Bjork-Shiley Convexo-Concave Heart Valve Single-Leg Separations from Acoustic Measurements*, Lawrence Livermore National Laboratory report UCRL-JC-119858, 1994.

9.2 Statistical Feature Selection

[20] P. A. Devijver, and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall, Englewood Cliffs, 1982, p. 448.

[21] P. M. Narendra and K. Fukunaga, *A Branch and Bound Algorithm for Feature Subset Selection*, IEEE Transactions on Computers, Vol. C-26, No. 9, September 1977, pp. 917-922.

[22] Anthony N. Mucciardi and Earl E. Gose, *A Comparison of Severn Techniques for Choosing Subsets of Pattern Recognition Properties*, IEEE Transactions on Computers, Vol. C-20, No. 9, September 1971.

[23] Roberto Battiti, *Using Mutual Information for Selecting Features in Supervised Neural Net Learning*, IEEE Trans. Neural Networks, Vol. 5, No. 4, July 1994.

[24] Christos H. Papadimitriou and Kenneth Steiglitz, *Chapter 18: Branch-and-Bound and Dynamic Programming*, in the book *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., 1982, pp. 433-453. ISBN 0-13-152462-3

9.3 Information Theory and the Hellinger Distance

[25] Claude E. Shannon and Warren Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, Chicago, London, Fifth Printing, 1972. ISBN 0-252-72548-4

[26] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, Wiley, 1991. ISBN 0-471-06259-6

[27] B. E. Bissinger, R. L. Culver, and N. K. Bose, *Minimum Hellinger Distance Based Classification of Underwater Acoustic Signals*, Acoustical Society of America Meeting, 2009.

[28] B. Bissinger, *Minimum Hellinger Distance Based Classification of Underwater Acoustic Signals*, Master of Science Thesis, The Pennsylvania State University, 2009.

[29] B. Bissinger, *Minimum Hellinger Distance Based Classification of Underwater Acoustic Signals*, Journal of the Acoustical Society of America, ASA Meeting, San Antonio, TX, October 2009.

[30] R. Beran, *Minimum Hellinger Distance Estimates for Parametric Models*, Anals of Statistics, Vol. 5, No. 3, 1977, pp. 445-463.

[31] A. Cutler and O. Cordero-Brana, *Minimum Hellinger Distance Estimation for Finite Mixture Models*, Journal of the American Statistical Association, Vol. 91, No. 436, 1996, pp. 1716-1723.

[32] L. Giet and M. Lubrano, *A Minimum Hellinger Distance Estimator for Stochastic Differential Equations: An Application in Statistical Inference for Continuous Time Interest Rate Models*, Computational Statistics and Data Analysis, Vol. 52, No. 6, Feb. 2008, pp. 2945-2965.

[33] B. G. Lindsay, *Efficiency vs. Robustness: the Case for Minimum Hellinger Distance and Related Methods*, Annals of Statistics, Vol. 22, No. 2, 1994, pp. 1081-1114.

[34] D. Donoho and R. Liu, *The Automatic Robustness of Minimum Distance Functionals*, Annals of Statistics, Vol. 16, 1988, pp. 552-586.

9.4 Statistical Pattern Recognition

[35] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, Second Edition, Wiley, 2001.

[36] D. G. Stork, E. Yom-Tov, *Computer Manual in MATLAB to Accompany Pattern Recognition*, Second Edition, Wiley-Interscience, 2004.

[37] Sturges, H., *The Choice of Class-Interval*, J. Amer. Statist. Assoc., Vol. 21, pp. 65-66.

[38] R. J. Hyndman, *The Problem with Sturges' Rule for Constructing Histograms*, http://robjhyndman.com/papers/sturges.pdf, July 5, 1995.

[39] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

[40] T. Hastie,R. Tibshirani,J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.

[41] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Third Edition, Elsevier, 2006.

[42] T. Y. Young and K-S Fu, *Handbook of Pattern Recognition and Image Processing*, Academic Press, 1986.

[43] R.M. Welch, K. Kuo, S.K. Sengupta, *Cloud and Surface Textural Features in Polar Regions*, IEEE Trans. Geoscience and Remote Sensing, vol. 28, No. 4, pp. 520-528, July 1990.

[44] R.P. Lippmann, *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, April, 1987, pp. 4-22.

[45] M. K. Hu, *Visual Pattern Recognition by Moment Invariants*, IRE Trans. Info. Theory, vol. IT-8, 1962, pp. 179-187.

[46] D. E. Rumelhart, J. L. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I and II*, Cambridge, MA: MIT Pr., 1986, p. 611.

[47] D. J. Hand, *Discrimination and Classification*, New York, L. Wiley and Sons, 1981, p. 218.

[48] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd Ed., Academic Press, 1990.

[49] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. 1, Addison Wesley, 1992.

[50] M. D. Richard, R. P. Lippmann, *Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities*, Neural Computation, vol. 3, pp. 461-483, 1991.

[51] M. T. Musave, K. H. Chan, D. M. Hummels, and K. Kalantri, *On the Generalization Ability of Neural Network Classifiers*, IEEE Trans. Pattern Recognition and Machine Intelligence, vol. 16, no. 6, June 1994.

[52] P. A. Devijver and J. Kittler, *Pattern Recognition Theory and Applications*, in Advances in Statistical Pattern Recognition, by A. K. Jain, Springer Verlag, 1987.

[53] E. Waltz and J. Llinas, *Multisensor Data Fusion*, Artech House, 1990.

[54] D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion*, Artech House, 1992.

[55] R. T. Antony, *Principles of Data Fusion Automation*, Artech House, 1995.

[56] B. V. Dasarathy, *Decision Fusion*, IEEE Computer Society Press, 1994.

[57] W.K. Pratt, *Digital Image Processing, 2nd Edition*, Wiley, 1978, pp. 559-561.

[58] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[59] R. C. Gonzales, and Wintz, Paul., *Digital Image Processing*, second edition. Addison-Wesley Publishing Company, 1987.

[60] E. L. Hall, *Computer Image Processing and Recognition*, Academic Press, 1979.

9.5 PDF Estimation and the Probabilistic Neural Network (PNN)

[61] D.E. Specht, *Probabilistic Neural Networks*, Neural Networks, Vol. 3, pp.109-118, 1990.

[62] D. F. Specht, *Enhancements to Probabilistic Neural Networks*, Proceedings, IEEE Intn'l Joint Conf. Neural Networks (IJCNN), 1992.

[63] P.V. Nageswara Rao, R Uma Devi, DSVGK Kaladhar, G.R. Sridhar, A. A. Rao, *A Probabilitstic Neural Network Appoach for Protein Superfamily Classification*, Journal of Theoretical and Applied Information Technology, Vol. 6, No. 1, 2005-2009, pp. 101-105.

[64] B. W. Silverman, *Density Estimation for Statisitcs and Data Analysis*, Chapman Hall/CRC, 1986.

[65] D. F. Specht, *Probabilistic Neural Networks*, Neural Networks, Vol. 3, pp. 109-118, 1990.

[66] J. Bibb Cain, *An Improved Probabilistic Neural Network and its Performance Relative to Other Models*, SPIE Vol. 1295 Applications of Arificial Neural Networks, 1990.

[67] D. F. Specht, *A General Regression Neural Network*, IEEE Trans. Neural Networks, Vol. 2, No. 6, November 1991.

[68] D. F. Specht, *Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification*, IEEE Trans. Neural Networks Vol. 1, No. 1, March 1990, pp. 111-121.

[69] T. Cacoullos, *Estimation of a Multivariate Density*, Annals of the Institute of Statistical Mathematics (Tokyo), vol. 18(2), 1966, pp. 179-198.

[70] V. K. Murthy, *Estimation of Probability Density*, Annals of Mathematical Statistics, vol. 36, 1965, pp. 1027-1031.

[71] V. K. Murthy, *Nonparametric Estimation of Multivariate Densities with Applications*, in P.R. Krishnaiah (Ed.), Multivariate Analysis, New York: Academic Press, 1966, pp. 43-58.

[72] E. Parzen, *On Estimation of a Probability Density Function and Mode*, Annals of Mathematical Statistics, vol. 33, 1962, pp. 1065-1076.

9.6 Bootstrap Algorithms

[73] A. M. Zoubir and D. R. Iskander, *Bootstrap Techniques for Signal Processing*, Cambridge U. Press, ISBN-13 978-0-521-03405-0, 2004.

[74] A. M. Zoubir and B. Boashash, *The Bootstrap and its Application in Signal Processing*, IEEE Signal Processing Magazine, January 1998.

[75] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall/CRC, 1998.

[76] B. Efron and R. Tibshirani, *Cross-Validation and the Bootstrap: Estimating the Error Rate of a Prediction Rule*, Unknown citation, circa 1993.

9.7 Probability, Random Variables, Stochastic Processes and Estimation

[77] J. V. Candy, *Model-Based Signal Processing*, IEEE Press and Wiley-Interscience, ISBN-13 978-0-471-23632-0 (cloth), ISBN-10 0-471-23632-2 (cloth), 2006.

[78] H. W. Sorenson, *Parameter Estimation,* Marcel Decker, Inc., 1980.

[79] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, Fourth Edition, McGraw-Hill, 2002.

[80] A. Papoulis, *Probability, Random Variables and Stochastic Processes,* McGraw-Hill, 1965.

[81] A. Jazwinski, *Stochastic Processes and Filtering Theory,* Academic Press, New York, NY, 1970.

9.8 Detection Theory and CFAR Detection

[82] H. L. Van Trees, *Detection, Estimation and Modulation Theory, Part I Detection, Estimation and Linear Modulation Theory*, John Wilery and Sons, 1968.

[83] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*, Kluwer Academic Publishers, The Netherlands, March 16, 2004.

[84] A. Whalen, *Detection of Signals in Noise* Academic Press, 1971.

[85] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume II Detection Theory*, Prentice Hall, 1998.

[86] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1965.

[87] X. Yu, I. S. Reed, A. D. Stocker, *Comparative Performance Analysis of Adaptive Multispectral Detectors*, IEEE Trans. Signal Processing, Vol., 41, No. 8, 1993.

[88] I. S. Reed and X. Yu, *Adaptive Multiple-Band CFAR Detection of an Optical Pattern with Unknown Spectral Distribution*, IEEE Trans. Acoustics, Speech and Signal Processing, Vol. 38, No. 10, October 1990.

9.9 Mathematics

[89] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statisitics*, Macmillan Publishing Co., 1978.

[90] G. Bachman, L. Narici, *Functional Analysis,* Academic Press, 1966.

[91] B. Noble, *Applied Linear Algebra,* Prentice-Hall, 1969.

[92] R. V. Churchill, *Complex Variables and Applications*, McGraw-Hill, 1960.

[93] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, Cambridge University Press, 1986.

[94] F. A. Acton, *Numerical Methods That Work*, Harper and Row, 1970.